Guía de administración de red de Debian GNU/Linux

Jorge Juan Chico <jjchico@imse.cnm.es>
Paulino Ruiz–de–Clavijo Vázquez <paulino@imse.cnm.es>

Versión 0.4 (3 julio 2002)

Resumen

Este documento es una guía de introducción a la configuración y administración de red de Debian GNU/Linux correspondiente a la versión 3.0 (woody) de la distribución.



Nota de Copyright

Índice General

Capítulo 1

Introducción

Este documento comprende los aspectos básicos de la configuración y administración de lo que podría ser un pequeño servidor en Internet, junto con los procedimientos para poner en marcha diferentes servicios como correo electrónico, páginas web, FTP anónimo, etc.

Este documento pretende ser una guía práctica y concisa para la configuración y administración de la red en un sistema Debian GNU/Linux. Dado el carácter conciso de este documento, ningún tema se tratará con profundidad, pero sí de forma suficiente para configurar y mantener los servicios más comunes sin necesidad de consultar documentación adicional.

Para profundizar más en cualquiera de los temas se recomiendan especialmente los documentos "HOWTO" (CÓMO) que pueden encontrarse en `www.linuxdoc.org`. Estos documentos son guías prácticas sobre temas concretos, entre ellos hay varios dedicados a redes, como el "The Linux Networking Overview HOWTO" y el "Linux Networking–HOWTO", que proporcionan una excelente información introductoria sobre las capacidades de red de Linux y la configuración básica, además de numerosos enlaces más especializados.

En general, debe tenerse en cuenta que tanto los documentos "HOWTO" como la documentación general de los programas no suele estar referida a una distribución de Linux concreta. De hecho, en la distribución Debian, muchas de las tareas de configuración e instalación descritas en dicha documentación se hacen de forma automática, sin que el usuario/administrador tenga que preocuparse por los detalles. Uno de los objetivos de la presente guía es, precisamente, describir los mecanismos concretos que proporciona la distribución Debian GNU/Linux para la configuración de la red y la instalación de los servicios más comunes.

Capítulo 2

Introducción a Internet

En este capítulo damos una breve introducción al funcionamiento de Intenet y a los elementos de configuración de un ordenador que opere en internet.

2.1 Redes de área local

Para la interconexión física de varios ordenadores se emplea algún tipo de tecnología de red de área local (LAN). La tecnología más común con diferencia son las redes Ethernet. Para construir una red de este tipo, cada ordenador debe poseer una tarjeta de red (interfaz de red en lenguaje más técnico) que se inserta dentro del ordenador y algún dispositivo de conexión externo. Este dispositivo externo suele ser un concentrador o *hub* al que se conectan las tarjetas de red de los diferentes ordenadores mediante un cable específico (cable de pares trenzados con conectores RJ45, en la mayoría de los casos). Otra modalidad más antigua pero todavía en uso, no emplea ningún dispositivo externo, sino que un cable coaxial une en forma de cadena todos los ordenadores a través de sus tarjetas de red.

De uno u otro modo, cada ordenador conectado a la red puede intercambiar datos con los demás en forma de paquetes que se insertan en la red local. Para organizar este intercambio, cada ordenador dispone de una *dirección* única que viene grabada de fábrica en cada tarjeta, por lo que se suele llamar *dirección física*. La dirección física en las redes Ethernet se compone de un conjunto de 6 bytes (48 bits) que se suelen representar en hexadecimal. Por ejemplo: 08:00:2b:4c:59:23.

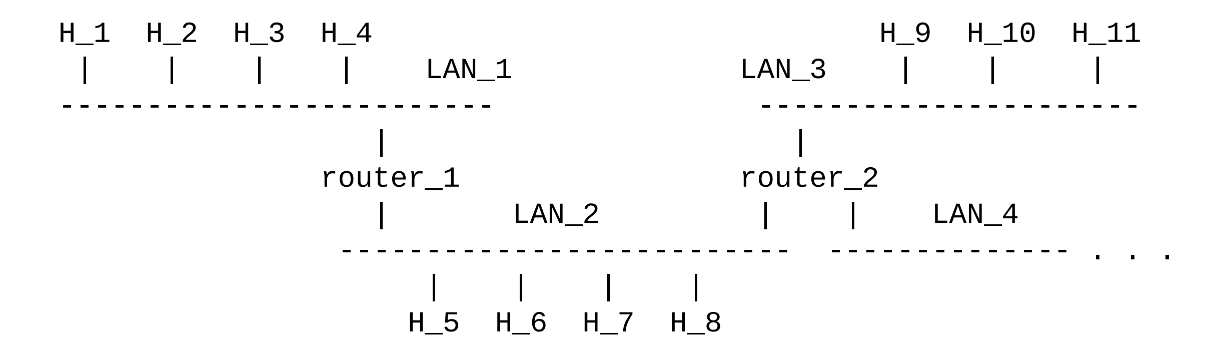
Otro mecanismo muy común para conectarse a una red es mediante una conexión *punto a punto* a otro ordenador, por ejemplo mediante el uso de un *módem*. En este caso, la red local está formada por sólo dos ordenadores, los que se encuentran a ambos extremos del cable o de la línea telefónica. La tecnología equivalente a las redes Ethernet para este tipo de conexiones se denomina PPP (*Point to Point Protocol*)

2.2 Internet

La tecnología de redes de área local a secas limita el intercambio de información a la propia red local, que no pueden extenderse más allá de las dimensiones de un laboratorio, un edificio o a lo sumo un campus,

y se componen de un número del orden de cien ordenadores. Para permitir la conexión de ordenadores en áreas más amplias se desarrolló Intenet. Básicamente, Intenet proporciona un mecanismo para la interconexión de redes de área local entre sí, y para la transferencia de información entre ellas. De este modo es posible el intercambio de información entre ordenadores muy distantes, conectados a redes de área local diferentes.

De esta forma, Internet se compone de un conjunto heterogéneo de redes de área local repartidas por todo el mundo, interconectadas entre ellas por máquinas especiales denominadas *routers* que se encargan de traspasar los paquetes de datos de unas redes a otras según sea necesario, como se muestra en el ejemplo.



2.3 Elementos de configuración de un ordenador en Internet

Cada ordenador conectado a internet necesita conocer cierta información sobre si mismo y sobre la propia red para que pueda funcionar correctamente. Esta información depende de si se trata de un ordenador común (*host*) o un *router*.

2.3.1 Elementos de configuración de un host

Dirección IP y máscara de subred

La dirección IP está formada por cuatro bytes (32 bits) e identifica a un ordenador de forma unívoca en Internet. Un ordenador puede poseer varias direcciones IP si dispone de varios interfaces de red, como en el caso de un router, asignando una dirección IP a cada interfaz. La forma habitual de representar las direcciones IP es escribiendo cada byte por separado en decimal, separándolos por puntos, por ejemplo: “150.214.141.122”.

En Internet se definen subredes agrupando cierto número de direcciones IP contiguas. Todas las direcciones dentro de una subred comparten un cierto número de bits. Los bits de la dirección que son propios de la subred se identifican especificando la *máscara de subred*, que esta compuesta por 32 bits, donde se ponen a 1 los correspondientes a la subred y a 0 los propios del host. En el siguiente ejemplo:

```
Dirección IP:      150.214.141.122
Máscara de subred: 255.255.255.0
Parte de la red:   150.214.141.
```

```
Parte del host:               .122
----------------------------------
Dirección subred:  150.214.141.0
Dir. broadcast:    150.214.141.255
```

la máscara de subred tiene los primeros 24 bits a 1 y el resto a 0. Esto indica que los primeros 24 bits son comunes a todos los ordenadores de la subred, y que sólo los últimos 8 bits están disponibles para distinguir entre ordenadores dentro de la subred. Cada ordenador debe conocer la máscara de la subred en que se encuentra, pues por convenio todos los ordenadores en la misma subred deben pertenecer a la misma red local y por tanto son accesibles directamente sin necesidad de pasar por un router.

En cada subred hay dos direcciones IP especiales que no se emplean para ningún host. Una es la dirección de la propia subred, que se construye con todos los bits de la parte del host a 0, y la otra es la dirección de broadcast, que se construye con todos los bits de la parte del host a 1. La dirección de broadcast hace referencia a todos y cada uno de los hosts de la subred al mismo tiempo.

Puerta de enlace (*Gateway*)

Un ordenador puede enviar paquetes de datos de forma directa sólo a ordenadores que se encuentren en su propia red local. El ordenador sabe que el destino está en su misma red local si la dirección IP del ordenador de destino posee la misma parte de la red que su propia dirección IP. Si la dirección IP de destino no está en su misma red local, el paquete debe ser enviado a un router para que se encargue de su transporte hasta el destino. Todo ordenador en internet tiene definido un router por defecto o puerta de enlace (Gateway) al que se envían todos los paquetes que van fuera de la red local. Esto permite la conexión con el resto del mundo, aparte de la propia subred. Aunque no es necesario, por convenio se suele asignar la dirección de host 1 al router por defecto, por ejemplo: “150.214.141.1”.

Servicio de nombres

Aunque las direcciones IP son útiles para identificar los ordenadores conectados a internet, resulta más conveniente su identificación mediante nombres, por ejemplo, es más fácil recordar el nombre “www.debian.org” que la dirección IP “198.186.203.20”. Además, los nombres facilitan una flexibilidad adicional porque pueden reasignarse fácilmente a otras direcciones por cambio de localización de un servidor o avería, e incluso puede resultar conveniente asignar varios nombres a un mismo ordenador. En este punto baste decir que los nombres se organizan en una jerarquía de dominios y subdominios. El nombre de un ordenador se compone del nombre propio de la máquina seguido por los nombres de los subdominios a los que pertenece dentro de la jerarquía hasta llegar a uno de los dominios globales como *com*, *org*, o *es*. Lo que nos interesa saber aquí es que podemos utilizar nombres de dominio en vez de direcciones siempre que indiquemos en la configuración de Internet las direcciones de unos servidores de nombres que se encargan de traducir los nombres a las direcciones IP correspondientes. Normalmente, pueden indicarse varios servidores de nombres por si alguno de ellos falla.

Nombre de la máquina y del dominio

Como parte del servicio de nombres, un host debe conocer cual es su propio nombre y el dominio al que pertenece. Éstos no son arbitrarios, sino que han de estar registrados en el servidor de nombres que corresponda a nuestro dominio. Por ejemplo:

```
Nombre del host  klecker
Dominio:         debian.org
------------------------------
Nombre completo: klecker.debian.org
```

2.3.2 Elementos de configuración adicionales en un router

Si el ordenador en cuestión se conecta simultáneamente a más de una red y, por tanto, puede actuar como router, la configuración de Internet debe contemplar además los siguientes puntos:

- Debe asignarse una dirección IP por cada interfaz físico (conexión de red).
- Debe definirse una tabla de rutas que indique por que interfaz deben enviarse los paquetes según su dirección de destino, puesto que ahora existe más de una conexión a la red.
- Si la máquina va a actuar como router (caso más común), debe configurarse para que haga el reenvío de los paquetes recibidos por un interfaz hacia el interfaz de salida correspondiente (*forwarding*).

2.3.3 Direcciones IP especiales

Aparte de las direcciones IP que identifican una subred o bien una dirección de broadcast, existen otras direcciones especiales. Las más interesantes son aquellas destinadas a redes que emplean lo protocolos de Internet (redes IP) pero que no están directamente conectadas a la red global, esto es, no pertenecen a la Internet. Estas redes se denominan a menudo redes privadas o intranets. En principio, en redes aisladas de Internet podrían usarse las direcciones IP que el administrador considerara oportuno siempre que fueran localmente distinguibles, pero por razones de seguridad y organización se definen rangos de direcciones IP para estas redes. Estos son los siguientes:

```
-----------------------------------------------------------
|       DIRECCIONES RESERVADAS PARA REDES PRIVADAS        |
-----------------------------------------------------------
| Clase   | Máscara       | Direcciones de red            |
| de red  |               |                               |
-----------------------------------------------------------
|    A    | 255.0.0.0     | 10.0.0.0    - 10.255.255.255  |
|    B    | 255.255.0.0   | 172.16.0.0  - 172.31.255.255  |
|    C    | 255.255.255.0 | 192.168.0.0 - 192.168.255.255 |
-----------------------------------------------------------
```

Por razones obvias, nunca debería aparecer tráfico en Internet correspondiente a estas direcciones.

2.4 Puertos y servicios

Mediante un nombre o una dirección IP puedo localizar un ordenador concreto en Internet, pero además necesito un mecanismo para poder conectar a un servicio concreto dentro de ese ordenador (correo electrónico, páginas web, FTP, etc.). La conexión a un servicio concreto, o mejor dicho, a un proceso concreto que se ejecute en un ordenador se realiza especificando un número de puerto. Así, por ejemplo, los programas que se encargan de enviar el correo electrónico "escuchan" en el puerto 25 y los servidores web en el puerto 80. Estos puertos son conocidos por los clientes correspondientes (lector de correo electrónico o navegador web).

Existen muchos servicios conocidos que tienen asignados puertos fijos, por ejemplo:

```
Servicio    Puerto     Descripción
-------------------------------------------------
ftp         21         transferencia de ficheros
ssh         22         conexión remota segura
telnet      23         conexión remota
smtp        25         transferencia de correo
finger      79         identificación de usuarios
www         80         WEB
pop3        110        servicio de correo
sunrpc      111        portmapper
```

Otros servicios estándar como el sistema de ficheros en red (NFS) no poseen un puerto fijo, sino que se les asigna dinámicamente por un servicio especial, el *portmapper*.

Los puestos 1 a 1024 se denominan *privilegiados* y sólo pueden ser empleados por procesos del administrador (*root*). El resto de los puertos están disponibles y se asignan dinámicamente según es necesario. Por ejemplo, si usamos un navegador web en el ordenador H1 para conectar a un servidor web en el ordenador H2, se asignará un puerto disponible en H1 a nuestro navegador web, por ejemplo el 1048 y tendremos una conexión desde el puerto 1048 de H1 al puerto 80 de H2:

```
Navegador web <--> H1:1048 <-----------> H2:80 <--> Servidor web
```

Capítulo 3

Configuración de la red

En este apartado se describe cómo configurar la conexión a la red de un ordenador. Las dos tareas básicas que conlleva la configuración de la red son:

- Configuración de la interfaz de red: tarjeta de red o conexión punto a punto.
- Configuración de TCP/IP (Protocolos de Internet): dirección IP, máscara de subred, etc.

Empezaremos por la configuración para redes Ethernet y los parámetros para la conexión a Internet y luego veremos como configurar conexiones punto a punto mediante módem.

3.1 Configuración de una tarjeta Ethernet

La configuración de las tarjetas Ethernet consiste básicamente en cargar el *driver* apropiado para la tarjeta o tarjetas de que dispongamos. Los drivers de las tarjetas de red se cargan como cualquier otro módulo del kernel (ver `modconf(8)`). El número de tarjetas de red soportadas por Linux es considerable: prácticamente todas las tarjetas de los fabricantes importantes están soportadas (ver el *Ethernet HOWTO* en `http://www.linuxdoc.org/HOWTO/Ethernet-HOWTO.html`).

Si disponemos de una tarjeta PCI, casi seguro que bastará con insertar el módulo correspondiente en el kernel sin indicar ningún parámetro adicional. En caso de que se trate de una tarjeta ISA es recomendable conocer previamente la interrupción (irq) y la dirección del puerto de entrada/salida empleado por la tarjeta (io) para indicárselo al programa modconf a la hora de instalar el driver.

En el caso de tarjetas Ethernet, cada módulo cargado asignará un nombre de dispositivo a la tarjeta, empezando por *eth0* y siguiendo con *eth1*, etc. Esto debe en equipos con varias tarjetas.

3.2 Configuración de TCP/IP

Aquí veremos cómo configurar los datos descritos en el apartado anterior.

3.2.1 Dirección IP, máscara y Gateway

Esta información se configura para todos los interfaces de red en el fichero `/etc/network/interfaces`. A continuación mostramos un ejemplo:

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
# automatically added when upgrading
auto lo
iface lo inet loopback

# The first network card - this entry was created during the Debian in-
stallation# (network, broadcast and gateway are optional)
# automatically added when upgrading
auto eth0
iface eth0 inet static
        address 150.214.141.122
        netmask 255.255.255.0
        network 150.214.141.0
        broadcast 150.214.141.255
        gateway 150.214.141.1
```

Las líneas que comienzan con "#" son comentarios. Las dos primeras líneas "reales" configuran una interfaz de red virtual, el lazo local (*local loopback*) que siempre ha de estar presente y que se configura automáticamente al instalar el sistema. El resto de líneas configuran los parámetros para una tarjeta Ethernet: dirección IP (*address*), máscara de subred (*netmask*) y dirección del router por defecto (*gateway*). Las directivas *network* y *broadcast* son opcionales y especifican la dirección de la subred y la dirección de broadcast. En este caso tienen los valores por defecto, con todos los bits de la parte del host a 0 para la dirección de la subred y a 1 para la dirección de broadcast. Las directivas *auto* indican que los interfaces mencionados deben configurarse automáticamente al arrancar el sistema.

En la página de manual `interfaces(5)` y en el fichero `/usr/share/doc/netbase/examples/interfaces` pueden encontrarse ejemplos de configuración de diferentes tipos de interfaces.

3.2.2 Nombre de la máquina

El nombre de la máquina se configura en el fichero `/etc/hostname`, cuyo contenido es simplemente el nombre de la máquina, sin indicación del dominio. De este fichero es leído el nombre durante el arranque del sistema. Para cambiar el nombre inmediatamente se emplea el comando `hostname(1)` indicando como argumento el nuevo nombre:

```
# hostname figaro
```

o bien, una vez modificado el fichero `/etc/hostname` puede ejecutarse a mano el *script* que inicial el nombre de la máquina:

```
# /etc/init.d/hostname.sh start
```

3.2.3 Dominio y servidores de nombres

El dominio y los servidores de nombres se configuran en el fichero `/etc/resolv.conf`. Un ejemplo de este fichero es el siguiente:

```
search dte.us.es fie.us.es
nameserver 150.214.186.69
nameserver 150.214.130.15
nameserver 150.214.4.34
```

donde se indica que el dominio principal es *dte.us.es*. Esto significa que si se indica un nombre de host sin especificar dominio, se asumirá que éste es *dte.us.es*. En este ejemplo se indica un nombre de dominio adicional, que se usará cuando no se encuentre el host en el dominio principal. Por ejemplo, si iniciamos una conexión indicando como nombre de la máquina de destino "figaro", se intentará la conexión a la máquina llamada "figaro.dte.us.es". En caso de que esta máquina no exista, se intentará una conexión a "figaro.fie.us.es" y si esta también falla se producirá un mensaje de error.

En las líneas siguientes se indican las direcciones de varios servidores de nombres. Éstos serán consultados por orden cuando se necesite traducir un nombre a una dirección IP. Se tendrá en cuenta la información del primero que responda.

3.2.4 Definición de nombres locales

Es posible definir localmente la correspondencia entre nombres y direcciones IP. De esta forma se puede acelerar la traducción de nombres a direcciones para aquellos nombres que se utilicen frecuentemente, y se pueden asignar alias sencillos a máquinas de interés. Esta correspondencia se define en el fichero `/etc/hosts`. El formato es muy sencillo y se muestra en el ejemplo siguiente correspondiente a la máquina *figaro.dte.us.es*:

```
127.0.0.1       localhost
150.214.149.100 figaro.dte.us.es        figaro
150.214.141.122 lorenzo.fie.us.es       lorenzo
```

Las dos primeras líneas se crean por defecto en la instalación. La primera asigna el nombre estándar *localhost* a la dirección 127.0.0.1, que corresponde por definición a la dirección del lazo local. Esto quiere decir que siempre podemos hacer referencia a la propia máquina empleando el nombre *localhost*. La segunda línea define la correspondencia entre la dirección IP y el nombre de la propia máquina. De esta forma se evita tener que acceder a un servidor de nombres externo cuando se hace referencia a la propia máquina por su nombre. Puede observarse que se ha definido tanto el nombre completo con el dominio como un nombre abreviado. Es importante cambiar los datos de esta línea si en el futuro se cambia el nombre de la máquina o su dirección IP. La tercera línea ha sido añadida por el administrador por conveniencia, al tratarse de una máquina que se referencia frecuentemente.

El uso de este fichero resulta conveniente en los casos en que no se dispone de servidores de nombres para suplir en parte esta carencia.

3.3 Comprobación de la red

3.3.1 Existencia de conexión

La forma más sencilla de comprobar si tenemos conexión es usando el comando `ping(8)`. Este comando manda paquetes sucesivos a una máquina indicando su nombre o su dirección IP y mide el tiempo empleado por cada paquete en llegar a su destino y volver. Por ejemplo:

```
$ ping 150.214.141.1
PING 150.214.141.1 (150.214.141.1): 56 data bytes
64 bytes from 150.214.141.1: icmp_seq=0 ttl=255 time=2.3 ms
64 bytes from 150.214.141.1: icmp_seq=1 ttl=255 time=3.9 ms
64 bytes from 150.214.141.1: icmp_seq=2 ttl=255 time=1.9 ms
64 bytes from 150.214.141.1: icmp_seq=3 ttl=255 time=1.3 ms
64 bytes from 150.214.141.1: icmp_seq=4 ttl=255 time=0.9 ms
```

En caso de problemas de conexión, debemos probar primero con máquinas de nuestra misma red local que sepamos que están funcionando. Preferentemente debemos usar direcciones IP cuando estemos haciendo comprobaciones, pues el comando puede fallar simplemente porque no tengamos acceso a los servidores de nombres. El comando `ping` dispone de una serie de opciones para controlar el número y tamaño de paquetes enviados. Consultar su página de manual para más información.

3.3.2 Comprobación de servidores de nombres

A veces, un excesivo retraso en establecer una conexión cuando se especifica el nombre de una máquina remota puede significar un mal funcionamiento del servidor de nombres. Puede suceder que el servidor de nombres principal no funcione y se produzca un retraso hasta que la petición se manda al servidor secundario. Si no es posible establecer conexiones indicando nombres, pero sí indicando direcciones IP, entonces es que ninguno de los servidores de nombres configurados funciona.

Puede comprobarse si los servidores de nombres están accesibles usando el comando `ping` sobre los servidores configurados en el fichero `/etc/resolv.conf`. También pueden hacerse peticiones a los servidores de nombres para que traduzcan un nombre en una dirección IP o viceversa usando el comando `host(1)`. Por ejemplo:

```
$ host www.debian.org 150.214.4.34
www.debian.org       A 198.186.203.20
$ host 150.214.4.34
Name: erik.cica.es
Address: 150.214.4.34
```

Usado con un sólo argumento, consulta a los servidores de nombres definidos en `/etc/resolv.conf`, pero puede especificarse un servidor de nombres concreto indicando un segundo argumento. El comando `host` dispone de múltiples opciones adicionales para realizar todo tipo de consultas a los servidores de nombres. Para saber más sobre estas opciones puede consultar la página de manual del comando.

3.3.3 Estado de los interfaces de red

La configuración de los interfaces de red se puede comprobar con el comando `ifconfig(8)`. Aunque se encuentra en el directorio `/sbin`, cualquier usuario puede ejecutarlo para esta tarea. Por ejemplo:

```
$ /sbin/ifconfig
eth0
Link encap:Ethernet  HWaddr 00:4F:4E:05:FA:35
inet addr:150.214.141.122  Bcast:150.214.141.255  Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:9373307 errors:1801 dropped:993 overruns:80 frame:0
TX packets:8026804 errors:3583 dropped:0 overruns:0 carrier:7166
collisions:224583 txqueuelen:100
RX bytes:1764525259 (1682.7 Mb)  TX bytes:3841778389 (3663.8 Mb)
Interrupt:9 Base address:0x4000

lo
Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:1193232 errors:0 dropped:0 overruns:0 frame:0
TX packets:1193232 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:316189089 (301.5 Mb)  TX bytes:316189089 (301.5 Mb)
```

muestra la configuración de los interfaces activos, en este caso una tarjeta Ethernet (*eth0*) y el lazo local (*lo*) que siempre debe estar presente. Además se muestra otra información como número de errores registrados hasta el momento y cantidad de información recibida y transmitida. Si en consultas sucesivas vemos que el número de errores aumenta constantemente y que no hay conexión, es muy probable que exista un fallo físico: cable en mal estado, concentrador desconectado, etc.

El comando `ifconfig` también se emplea para configurar los interfaces de red, pero en la mayoría de los casos no es necesario su uso directo, ya que el proceso está automatizado en la distribución Debian mediante el uso del fichero `/etc/network/interfaces` descrito en apartados anteriores.

3.3.4 Comprobación de la tabla de rutas

En sistemas sencillos, esto es, con una única conexión a la red. La tabla de rutas es muy sencilla y casi nunca tendremos que consultarla, salvo para asegurarnos que el router por defecto (*gateway*)

es el adecuado. La tabla de rutas puede consultarse con el comando `route(8)` o con el comando `netstat(8)`, indicando la opción `-r`. Por ejemplo:

```
$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Ifa
localnet        *               255.255.255.0   U     0      0        0 eth
default         150.214.141.1   0.0.0.0         UG    0      0        0 eth
```

La información suministrada indica básicamente dos cosas: que la red local (*localnet*) es la que se obtiene aplicando la máscara 255.255.255.0 a la dirección IP de la máquina y que a ella se accede a través del interfaz eth0, y que el gateway para el resto de destinos (*default*) está en la dirección IP 150.214.141.1.

El comando `route` también se emplea para modificar "manualmente" la configuración de la tabla de rutas, pero en la mayoría de los casos esta tarea es realizada automáticamente por el sistema a partir de la información contenida en los ficheros de configuración de la red.

3.4 Activación/desactivación de interfaces

Los interfaces de red configurados en el fichero `/etc/network/interfaces` pueden activarse y desactivarse usando los comandos `ifup(8)` e `ifdown(8)` respectivamente. Para ello basta con indicar el nombre del interfaz después del comando. Por ejemplo:

```
# ifdown eth0
```

desactivaría el primer interfaz Ethernet. El uso de estos comandos es útil cuando queremos "cortar" temporalmente la conexión a red o bien queremos cambiar los parámetros de configuración. Para ello, modificaremos en el fichero `/etc/network/interfaces` los parámetros pertinentes y luego ejecutaremos sucesivamente los comandos `ifdown` e `ifup` indicando el nombre del interfaz de red que hemos modificado. Por ejemplo:

```
# ifdown eth0 ifup eth0
```

Podemos comprobar que los cambios han tenido efecto usando el comando `ifconfig` como se describió anteriormente.

Los comandos `ifdown` e `ifup` se encargan, además de configurar los interfaces, de establecer las rutas necesarias y otras tareas, en función de los parámetros indicados en el fichero `interfaces(5)`, por lo que no suele ser necesario ejecutar manualmente los comandos `ifconfig` o `route`.

3.5 Configuración de una conexión PPP (módem)

En este apartado veremos como configurar una conexión a un proveedor de Internet a través de módem. Para ello se emplea el protocolo PPP. Aunque aquí sólo veremos la parte de la configuración del cliente, Linux incorpora soporte PPP tanto para cliente como servidor, por lo que es posible montar todo tipo de conexiones punto a punto, tanto de forma directa como usando la línea telefónica. Una completa referencia sobre conexiones punto a punto con Linux puede encontrarse en el *PPP–HOWTO*.

3.5.1 Elegir un módem

Los modems se pueden clasificar básicamente en dos tipos: los modems reales, que suelen ser externos e incorporan un chip que realiza las funciones del módem; y los modems *software* o *winmodems*, que suelen ser internos, mucho más baratos, pero en los que las funciones del módem se realizan en su mayoría por software por parte del driver que corre en el ordenador. Linux funciona perfectamente con los primeros y se lleva mal con los segundos. La razón de esto último es que los fabricantes no proporcionan información suficiente para realizar los drivers que necesitan los modems software. En cualquier caso, los modems reales dan mejores resultados y no imponen una carga adicional a la CPU del ordenador. La mejor forma de asegurarse de que un módem es realmente un módem es comprando un módem externo.

En este apartado supondremos que se tiene un módem externo conectado a un puerto serie, por ejemplo el `/dev/ttyS1`.

3.5.2 Configuración con `pppconfig`

La configuración de la conexión a un proveedor de Internet en Debian es muy sencilla usando el programa `pppconfig(8)`. El programa permite definir conexiones a diferentes proveedores y nos pregunta todos los datos necesarios paso a paso. Lo único que se necesita es disponer de la información necesaria del proveedor y ejecutar el programa. A continuación mostramos un ejemplo de la información necesaria para la conexión a un proveedor de Internet.

Datos necesarios para la configuración

- Nombre del proveedor: El nombre por defecto es “provider” y es el proveedor de conexión por defecto. Debe usarse éste si sólo vamos a tener un proveedor. Ejemplo: “provider”.
- Servidor de nombres: con la mayoría de los casos puede elegirse un servidor dinámico, que será suministrado por el proveedor en la conexión. Si esto falla debemos contactar con el proveedor para que nos suministre la IP de un servidor estático. Ejemplo: “dynamic”.
- El método de autentificación es “PAP” en la mayoría de los casos. Ante la duda, contactar con el proveedor. Ejemplo: “PAP”.
- Nombre de usuario en nuestra cuenta del proveedor. Ejemplo: “050@alehop”.

- Clave del usuario (password). Ejemplo: “gratis”
- Velocidad del puerto serie del módem. En la mayoría de los casos podemos dejar la opción por defecto. Ejemplo: “115200”.
- Método de marcación. En la actualidad, prácticamente todos los teléfonos utilizan tonos. Ejemplo: “Tone”.
- Número de teléfono del proveedor. Ejemplo: “955000123”.
- Puerto serie al que se conecta el módem. Hay una opción para detectarlo automáticamente, en cuyo caso debemos encender el módem previamente si éste es externo. Si conocemos el puerto podemos especificarlo manualmente. Ejemplo: “/dev/ttyS1”.

Finalmente salvaremos los cambios. Existe un menú de opciones avanzadas que normalmente no tendremos que modificar.

3.5.3 Activar/desactivar conexiones PPP

Para activar o desactivar una conexión PPP se emplean los comandos `pon(1)` y `poff(1)` respectivamente. El comando `pon` a secas inicia la conexión al proveedor de nombre “provider” (proveedor por defecto), pero se puede indicar como parámetro el nombre de otro proveedor para la conexión, por ejemplo:

```
$ pon eresmas
```

De la misma forma funciona el comando `poff`, pero para la desconexión. Si no se especifica nombre de proveedor, se desconecta del que esté activo.

Para que un usuario pueda usar estos comandos ha de tener los permisos apropiados sobre los ficheros `/dev/ttyS*`. Esto se consigue fácilmente añadiendo el/los usuario/s adecuados al grupo *dialout*. Por ejemplo:

```
# adduser pedro dialout
```

Existen muchos programas gráficos para ejecutar automáticamente los comandos `pon`y `poff` u otros equivalentes. Por ejemplo, aplique *Monitor Módem* de la sección de red del escritorio GNOME es bastante fácil de configurar para esta función.

Capítulo 4

Control de servicios de red

En este capítulo veremos cómo activar los servicios de red, esto es, como iniciar y detener los diferentes servidores (web, FTP, etc.). Veremos que una forma de controlar los servicios es mediante *scripts* dedicados que se sitúan en el directorio `/etc/init.d`. Esto hace normalmente que los servicios se inicien al arrancar la máquina y permanezcan activos indefinidamente, en espera de conexiones, salvo que el administrador los detenga manualmente. La otra alternativa es configurar el programa `inetd` para que active los servicios *bajo demanda*, cuando se detecta actividad en el puerto correspondiente. También veremos cómo controlar y restringir de forma global el acceso a los servicios en función del origen de la conexión. Finalmente daremos algunos consejos generales sobre seguridad de servicios en Internet.

4.1 Servicios independientes (*stand alone*)

Una de las formas de controlar el inicio y parada de los servicios de red es mediante el uso de *scripts* situados en el directorio `/etc/init.d`. De hecho, en este directorio se encuentran todos los scripts que intervienen en el arranque y parada del sistema. Todos los scripts que aparecen allí funcionan de forma similar: se deben ejecutar indicando un sólo parámetro que puede ser `start` o `stop` y, opcionalmente, otros como `restart`, `reload` o `force-reload`. El significado de estos parámetros es bastante obvio. Por ejemplo, un sistema Debian que tenga el servidor web Apache instalado dispondrá de un script llamado `apache` en el directorio `/etc/init.d`. Para detener el servidor basta con ejecutar:

```
# /etc/init.d/apache stop
```

De la misma forma, si lo que queremos es simplemente reiniciar el servidor tras haber modificado sus ficheros de configuración, bastaría con hacer:

```
# /etc/init.d/apache restart
```

La forma de iniciar estos servicios durante el arranque del sistema es mediante la creación de enlaces simbólicos que apunten hacia los scripts en los directorios `/etc/rc?.d`. Donde el signo “?” representa alguno de los *runlevels* configurados, normalmente el 2. Por ejemplo, la existencia del enlace:

```
/etc/rc2.d/S91apache -> ../init.d/apache
```

indica que el servidor web Apache será iniciado al arrancar el sistema. La “S” significa que script correspondiente será ejecutado con la opción `start` y el “91” indica un orden relativo entre todos los procesos a iniciar. De la misma forma, la presencia del enlace:

```
/etc/rc0.d/K20apache -> ../init.d/apache
```

indica que el servicio será detenido (“K") al parar la máquina, con número de orden 20. Más información sobre el proceso de inicio y parada del sistema puede encontrarse en la *Guía de administración de Debian GNU/Linux* y a partir del fichero `/etc/init.d/README`.

Cuando se instalan servidores a partir de paquetes de la distribución Debian, el propio mecanismo de instalación se encarga de crear estos scripts y los enlaces necesarios. Es más, el sistema de actualización de Debian es lo bastante inteligente para, en la mayoría de los casos, actualizar los servidores existentes e instalar los nuevos, parando y reiniciando los servicios según se necesite sin intervención del usuario. Este es uno de los puntos fuertes de la distribución Debian que la han hecho popular para su uso en sistemas servidor.

Cuando se instalan servidores no incluidos en la distribución Debian, el administrador es responsable de situar los scripts apropiados en los directorios correspondientes. Si el script de control no viene con el servidor a instalar, el propio administrador deberá crear uno, para lo que dispone de un modelo en el fichero `/etc/init.d/skeleton`. Para crear los enlaces adecuados, la distribución Debian proporciona un comando fácil de usar, por ejemplo, si queremos crear los enlaces para un servidor cuyo script tiene por nombre `local_server` y darle orden de secuencia 50, haríamos:

```
# update-rc.d local_server defaults 50
```

Si se omite el orden de secuencia, se toma 20 por defecto. Otras opciones del comando `update-rc.d(8)` pueden consultarse en su página de manual.

Finalmente daremos un par de consejos prácticos para cuando modifiquemos a mano la disposición de los scripts de iniciación:

- Si queremos que durante el arranque no se inicie un determinado servicio, pero no queremos desinstalar el servidor, mejor que borrar el enlace simbólico que lo inicial, podemos renombrarlo. Así recordaremos fácilmente qué había antes de nuestra modificación. Por ejemplo, para evitar que se inicie el servidor Apache, bastaría con entrar en el directorio `/etc/rc2.d` y hacer:

  ```
  # mv S91apache XS91apache
  ```

El hecho de que el enlace no comience por “S” impide que se ejecute durante el arranque del sistema

- Cuando instalemos nuestros propios scripts en el directorio `/etc/init.d` es conveniente darles nombres de forma que resulte fácil identificar que que estos scripts no pertenecen a la distribución, por ejemplo comenzando los nombres con “local", como en `local_webserver`.

4.2 Servicios activados bajo demanda: `inetd`

Otro mecanismo con que cuenta UNIX/Linux para activar servidores es la activación bajo demanda. En este caso, un solo servidor, el `inetd(8)`, “escucha” en muchos puertos simultáneamente a la espera de conexiones y cuando detecta actividad en alguno de estos puertos arranca el servidor correspondiente al servicio solicitado. Este sistema es conveniente por dos razones: evita tener en continuo funcionamiento un gran número de servidores que sólo se emplean ocasionalmente, y permite conceder o denegar el acceso en función del origen de la conexión antes de iniciar el servidor correspondiente.

Este sistema se emplea para muchos servicios estándar que se utilizan de forma ocasional y que consisten en conexiones generalmente de corta duración: telnet, ftp, correo electrónico, etc. Cuando se trata de servicios con una gran demanda de conexiones o de servidores cuyo tiempo de arranque es elevado, es preferible la iniciación en el arranque de estos servicios: servidor web, ftp anónimo, etc.

4.2.1 Configuración de `inetd`

`inetd` se configura editando el fichero `/etc/inetd.conf`, cuya página de manual puede consultarse en `inetd.conf(5)`. Cada línea del fichero sirve para iniciar un servicio. Las líneas que comienzan con “#” son comentarios y se ignoran. A continuación mostramos un ejemplo:

```
#:MAIL: Mail, news and uucp services.
smtp    stream  tcp   nowait   mail    /usr/sbin/exim exim -bs
#:STANDARD: These are standard services.
ftp     stream  tcp   nowait   root    /usr/sbin/tcpd /usr/sbin/in.ftpd
```

La primera línea sirve para iniciar el servidor de correo electrónico, en concreto el *agente de transporte de correo* (MTA) o servidor de correo saliente. El nombre del servicio es *smtp* y el programa a ejecutar es `/usr/sbin/exim`, que se ejecutará como usuario *mail*. Los parámetros adicionales tras el nombre del programa son la línea de comandos que se ejecutará para arrancar el servidor. La segunda línea sirve para activar el servidor para transferencia de ficheros (FTP). La diferencia más interesante con el anterior es que ahora el programa a ejecutar es `/usr/sbin/tcpd`, a pesar de que la línea de comandos ejecutará el servidor FTP (`in.ftpd`). La finalidad de esto es pasar el control inicialmente al programa `tcpd` antes de ejecutar la línea de comandos. La misión de `tcpd` es controlar el acceso a los servicios, como veremos un poco más abajo.

En la distribución Debian, el mantenimiento del fichero `/etc/inetd.conf` se hace de forma prácticamente automática. El sistema se encarga de instalar las líneas adecuadas cuando instalamos los

programas de la distribución que las necesitan, y de eliminarlas o desactivarlas cuando ya no se necesitan

El sistema guarda una lista de los nombres de servicios declarados que pueden usarse, por ejemplo, en el fichero `inetd.conf`. Esta lista se encuentra en el fichero `/etc/services`, cuya descripción se encuentra en la página de manual `services(5)`. El fichero muestra los nombres de los servicios y el protocolo o protocolos que emplea y el puerto asignado, por ejemplo:

```
...
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp      # SSH Remote Login Protocol
ssh             22/udp      # SSH Remote Login Protocol
telnet          23/tcp
finger          79/tcp
www             80/tcp      # WorldWideWeb HTTP
www             80/udp      # HyperText Transfer Protocol
...
```

4.2.2 Reiniciar `inetd`

Cuando se hacen cambios en el fichero `/etc/inetd.conf` es necesario notificar este cambio al proceso `inetd` en ejecución para que sea releída la nueva configuración. Existen dos formas: la primera consiste en reiniciar el servidor `inetd` tal y como haríamos con cualquier servidor activado mediante un script en `/etc/init.d`, esto es:

```
# /etc/init.d/inetd restart
```

Esto detiene el proceso `inetd` actual e inicia uno nuevo, que leerá la nueva configuración. La segunda forma es quizá más cómoda y eficiente, ya que consiste en mandar una señal a proceso actual para que relea la configuración, sin necesidad de detenerlo. El nombre de la señal a enviar es “HUP” y puede enviarse con el comando `killall(1)`:

```
# killall -HUP inetd
```

4.2.3 Desactivar servicios controlados por `inetd`

Si en algún momento queremos desactivar alguno de los servicios controlados por `inetd` pero no queremos desinstalar el servidor correspondiente, podemos hacerlo simplemente comentando la línea adecuada en el fichero `/etc/inetd.conf`, añadiendo un sólo “#” al principio de la línea deseada. De esta forma el sistema sabrá que se trata de una modificación manual del usuario y no activará esta línea en el futuro.

Es importante no modificar líneas comentadas con la cadena “#<off>#", pues es la marca que emplea el sistema cuando un servicio se desactiva automáticamente. Es importante también no modificar las cabeceras de las secciones que comienzan con “#:*:".

Tras modificar el fichero /etc/inetd.conf debemos reiniciar el proceso inetd como se ha indicado anteriormente. También debemos tener en cuenta que cambiar la configuración de inetd para un servidor no significa que se detenga la copia de ese servidor actualmente en curso. Para ello tendríamos que matar el proceso. Por ejemplo, para detener exim arrancado desde inetd haríamos:

```
# killall exim
```

Esto mismo sirve para reiniciar un servicio controlado por inetd, ya que la próxima vez que haya actividad en el puerto del servidor eliminado, éste volverá a ejecutarse.

4.3 Control de acceso a servicios: *TCP wrapper*

Existe un mecanismo que permite controlar el acceso a los servicios en función del origen de la conexión. Este mecanismo se configura con los ficheros /etc/hosts.allow y /etc/hosts.deny. En estos ficheros se especifican unas sencillas reglas que permiten denegar o aceptar el acceso en función de los parámetros de la conexión deseada. En las páginas de manual hosts_access(5) y hosts_options(5) se describe en detalle las capacidades de estos ficheros. Aquí describiremos el funcionamiento básico y pondremos algunos ejemplos.

Este mecanismo de control es empleado por todos los servicios que hagan uso de la librería libwrap, como es el caso de algunos servidores como sshd(8) y de los servicios activados mediante inetd que son controlados por tcpd(8), que son la mayoría.

Los ficheros /etc/hosts.allow y /etc/hosts.deny consisten en una serie de secuencias de control de la forma:

```
daemon_list : client_list [ : shell_command ]
```

donde daemon_list es la lista de servidores (nombre del ejecutable) al que hace referencia la línea de control y client_list es la lista de clientes a los que se autoriza o deniega el acceso en esa línea de control. Opcionalmente se puede incluir una línea de comandos a ejecutar cuando se cumple una regla.

La lista de servidores puede ser uno o más nombres separados por espacios o por comas. Pueden usarse caracteres de sustitución (*,?) para referirse a un conjunto de nombres. También puede emplearse la palabra ALL para indicar “cualquier servidor". La lista de clientes es una lista de direcciones IP (ej. 150.214.141.122), subredes (ej. 150.214.141.), nombres de máquinas (fabio.fie.us.es), o subdominios (ej. .fie.us.es), separados por comas o espacios. En la lista de clientes también se pueden usar las palabras LOCAL para especificar conexiones locales (desde la misma máquina) y la palabra EXCEPT para hacer exclusiones de la lista, como veremos en algunos ejemplos más adelante.

El funcionamiento es el siguiente:

1. cuando un servicio arranca, se comprueba el contenido del fichero `/etc/hosts.allow` en busca de reglas apropiadas. Si alguna regla concuerda con las características de la conexión, se permite el acceso. Por ejemplo:

   ```
   # fichero /etc/hosts.allow

   in.tftpd: LOCAL, .fie.us.es sshd: ALL EXCEPT .crackers.com

   ipop3d: 212.79.132.243 150.214.142.
   ```

   La primera línea permite el acceso al servidor TFTP (`in.tftpd`) desde la propia máquina y desde cualquier máquina del dominio *fie.us.es*. La segunda permite la conexión desde cualquier lugar al servicios de *shell* seguro `sshd`, excepto desde el dominio *crackers.com*. La última línea permite el acceso al servidor POP desde una máquina concreta y desde la subred 150.214.142.0.

2. Si ninguna regla en `/etc/hosts.allow` concuerda con la conexión, se pasa a buscar reglas que concuerden en el fichero `/etc/hosts.deny`. Si alguna regla de este fichero concuerda, el acceso es denegado inmediatamente. Por ejemplo:

   ```
   # fichero /etc/hosts.deny

   in.ftpd: .fie.us.es EXCEPT mipc.fie.us.es
   ```

   La línea de este fichero niega el acceso al servicio FTP desde el dominio *fie.us.es*, excepto para la máquina *mipc.fie.us.es*.

Debe tenerse en cuenta que el fichero `hosts.allow` tiene preferencia sobre el `hosts.deny`, de modo que un acceso permitido en el primero nunca podrá ser negado en el segundo.

4.3.1 Políticas de seguridad

El control de acceso basado en los ficheros `hosts.allow` y `hosts.deny` suele emplearse para implementar dos políticas de seguridad típicas: *casi todo abierto* o *casi todo cerrado*.

Política de seguridad *casi todo abierto*

En este esquema, todo lo que no esté expresamente denegado está permitido. Se implementa dejando vacío el fichero `hosts.allow` y denegando el acceso a servicios concretos en el `hosts.deny`. Por ejemplo:

```
# fichero /etc/hosts.allow
# (aquí no ponemos nada)

# fichero /etc/hosts.deny
ALL: some.host.name, .some.domain
in.fingerd: other.host.name, .other.domain
```

Política de seguridad *casi todo cerrado*

En este esquema de seguridad, todo lo que no esté expresamente permitido, está denegado. Se implementa con una línea de tipo “`ALL: ALL`” en el fichero `hosts.deny` y permitiendo accesos concretos en el fichero `hosts.allow`. Por ejemplo:

```
# fichero /etc/hosts.deny
ALL: ALL

# fichero /etc/hosts.allow
ALL: LOCAL
ALL: .fie.us.es EXCEPT alumnos.fie.us.es
```

4.3.2 Nombres especiales

Ya hemos visto que el nombre `ALL` puede usarse para identificar cualquier servicio o cualquier origen. De la misma forma, `EXCEPT` sirve para introducir una lista de excepciones como hemos visto en los ejemplos anteriores.

Un nombre de servicio especial es `portmap` y hace referencia al proceso `portmap(8)`. La mayoría de los servicios que hemos visto tienen asignado un número de puerto fijo, relacionado en el fichero `/etc/services`. Pero algunos servicios de red no poseen puertos fijos asociados, sino que se accede a ellos por nombre. Estos servicios se denominan servicios RPC (*Remote Procedure Call*). Un proceso especial, el *Portmapper* (`portmap`) se encarga de traducir para los clientes el nombre al número de puerto que emplea en ese momento el servicio deseado. Cuando se controla el acceso al servicio `portmap` es necesario indicar los orígenes mediante direcciones IP, no valen nombres.

Un nombre especial de cliente es `PARANOID`. Hace referencia a cualquier ordenador cuyo nombre no coincida con su dirección IP, según la información que pueda recopilarse del servidor de nombres. Este es un mecanismo para protegerse de ante posibles conexiones por parte de máquinas que intentan usar las direcciones de otras.

4.3.3 Comandos asociados a reglas

Es posible asociar comandos y otras opciones a reglas de modo que el sistema puede ejecutar cualquier proceso ante, por ejemplo, una conexión no deseada. Los detalles pueden consultarse en la página de manual `hosts_options(5)`. El siguiente ejemplo muestra la forma de invocar un comando que informa por correo electrónico al administrador ante cualquier intento de conexión no permitido:

```
# fichero /etc/hosts.deny
ALL: ALL : severity mail.info : rfc931 5 : spawn \
      (echo "Conexión denegada\: %u@%h -> %d (%p)" | \
       /usr/bin/mail -s Conexión_denegada root miguel) &
```

Si, por ejemplo, el usuario *paco* intenta conectar con el servidor POP desde la máquina frodo.fie.us.es, y el acceso no está permitido en el fichero `/etc/hosts.allow`, el administrador recibirá un mensaje de correo electrónico como el siguiente, indicando el origen de la conexión, el servicio y el número de proceso del servidor:

```
Subject: Conexión_denegada

Conexión denegada: paco@frodo.fie.us.es -> ipop3d (2521)
```

4.4 Consejos sobre seguridad

Por desgracia, Internet no fue concebida en sus inicios para que fuera una red segura. Afortunadamente, con el tiempo han ido apareciendo alternativas seguras a muchos de los servicios clásicos. En cualquier caso, el administrador de un sistema servidor debe que tener presente en todo momento la seguridad que proporcionan los servicios que presta.

Cubrir todos los detalles de la seguridad de redes TCP/IP comprende material suficiente para escribir un libro completo, pero con un poco de sentido y siguiendo unas reglas generales es posible mejorar sustancialmente la seguridad de un sistema. A continuación daremos una serie de recomendaciones a tener en cuenta para mejorar la seguridad del sistema que administremos:

- No pensar que el nuestro sistema pasará inadvertido y que los ataques siempre les ocurren a los otros. Si prestamos un servicio en Internet, tarde o temprano alguien analizará nuestro sistema para buscar fallos de seguridad. Hay herramientas muy eficientes para hacer esto y puede hacerse ¡desde cualquier parte del mundo!

- No proporcionar servicios inseguros. Sobre todo, aquellos que permiten mandar *passwords* no encriptados por la red están totalmente prohibidos. Por ejemplo: telnet, rsh, rlogin, ftp no anónimo, pop, imap, xdm, etc. En muchos casos es posible emplear alternativas seguras: ssh, scp, pop3s, imaps, etc.

- No proporcionar servicios que no se usen o que no sean necesarios, por ejemplo: echo, ident, talk, finger, etc. No dejar el servidor correspondiente sin activar, desinstalarlo completamente.

- Usar siempre una política de *denegar por defecto* (casi todo cerrado). Así, en caso de ataque o intrusión, las vías estarán mucho mejor localizadas.

- No proporcionar (en páginas web, mensaje de login, etc.) información concreta o detallada sobre el sistema: versión del kernel, distribución instalada, programas instalados, etc. Esta información es vital para los que buscan agujeros de seguridad.

- Mantener actualizado el sistema. Continuamente se encuentran fallos de seguridad en los programas que rápidamente se hacen conocidos. Éstos se solucionan rápidamente, pero para ello tenemos que tener el sistema actualizado. En Debian se puede estar al día de los parches de seguridad incluyendo una línea como esta en el fichero `/etc/apt/sources.list`:

```
deb http://security.debian.org stable/updates main contrib non-free
```

El sistema debe actualizarse al menos una vez a la semana. Para obtener información puntual sobre problemas de seguridad detectados en Debian puede consultarse la página web de la Distribución (`www.debian.org`o suscribirse a la lista de difusión de errores de seguridad (*debian–security–announce*)

- Revisar los ficheros de registro (*logs*) periódicamente en busca de conexiones denegadas o direcciones de origen *extrañas*. En especial los ficheros auth.log, daemon.log, messages y syslog, todos ellos en `/var/log`.
- Proteger los ficheros y directorios con información sensible de la mirada de los usuarios comunes, como por ejemplo: `/var/log`, `/etc/ssh`, `inetd.conf`, `hosts.allow` y `hosts.deny`.
- Poner fecha de caducidad a las cuentas de usuario con el comando `chage(1)`. Una cuenta activa y desatendida es una puerta de entrada al sistema.
- Comprobar de vez en cuando los puertos TCP y UDP que están abiertos con los comandos "`netstat -lp`" y `nmap`.
- Si se cree necesario, instalar un filtro de paquetes (*firewall*) con una política de denegar todas las conexiones salvo a los servicios que deseemos prestar y que sepamos que son seguros.

Capítulo 5

Correo electrónico

Para una mayor claridad, vamos a considerar el caso en que queremos configurar servicios de correo electrónico en un servidor concreto llamado *frodo.lab.es* para una serie de usuarios. El servidor podrá usarse desde un cliente de correo remoto (como *Netscape* o *Microsoft Outlook*™ tanto para enviar correo (servidor de correo saliente) como para recibir correo (servidor de correo entrante). Los usuarios tendrán direcciones de correo del tipo *pedro@frodo.lab.es*, donde en la dirección aparece el nombre del servidor, pero también veremos cómo hacer para que nuestro servidor gestione el correo del dominio y poder usar direcciones del tipo *pedro@lab.es*

Para crear las cuentas de correo electrónico no hay que hacer nada especial. Cualquier usuario del sistema posee una cuenta de correo, por lo que basta con añadir un usuario al sistema para crearle una cuenta de correo.

5.1 Servidor de correo saliente: SMTP

El programa que se encarga del correo saliente es el servidor SMTP. En realidad estos programas actúan como auténticos *routers* para el correo electrónico, por lo que se llaman *Agentes de Transporte de Correo* (MTA). El MTA que instala por defecto la distribución Debian es `exim(8)`, pero hay otros disponibles como `qmail(8)` o el famoso `sendmail(8)`. Aquí explicaremos la configuración de `exim`.

5.1.1 Configuración inicial de `exim`

El programa `eximconfig(8)` se encarga de la configuración inicial de `exim`. Este programa se ejecuta en la instalación inicial de `exim` y puede ejecutarse posteriormente para modificar la configuración inicial. El objetivo de este programa es poner en marcha el servidor de correo electrónico con unas opciones iniciales. La configuración posterior del servidor se realizará haciendo las modificaciones apropiadas en el fichero de configuración (`/etc/exim/exim.conf`). El programa `eximconfig` pregunta una serie de parámetros que explicamos a continuación:

- Uso y situación del servidor: se presentan 5 opciones a elegir una. Elegimos la 1 para configurar un servidor de correo electrónico en Internet. En caso de instalaciones locales, sin conexión

permanente a Internet o cuando no queremos ser un servidor de correo electrónico, elegiremos la opción 4, que sólo hace distribución del correo local. Esto es necesario porque muchos mensajes de error generados por el sistema se distribuyen a través del correo local. En nuestro ejemplo, elegimos la 1.

- Ahora se nos pregunta el nombre "visible" de nuestro sistema, que es el que aparecerá como remite en los mensajes enviados. En nuestro caso este nombre es *frodo.lab.es*. Si quisiéramos disponer de un remite de la forma *lab.es* tendríamos que solicitar al administrador del servidor de nombres del dominio *lab.es* que configurara la máquina *frodo.lab.es* como servidor de correo del dominio *lab.es*. En este caso podríamos emplear *lab.es* como nombre visible del sistema. Lo importante aquí es saber que no podemos hacer esto sin la colaboración del servidor de nombres. En nuestro ejemplo, elegimos *frodo.lab.es*.

- A continuación se nos pregunta si nuestro sistema es conocido por otro nombre además del anterior. En nuestro ejemplo no hay nombres adicionales y elegiremos la opción por defecto: "none". En caso de cooperación con el servidor de nombres, si antes hemos indicado como nombre visible *lab.es*, aquí podríamos indicar además *frodo.lab.es*, de esta forma el sistema reconocerá como propio el correo electrónico enviado a ambos dominios.

- Ahora se nos informa que, en la configuración por defecto, nuestro agente de correo aceptará correo que venga de Internet con destino a *frodo* o bien correo generado localmente en *frodo* con destino a Internet, pero nuestra máquina no hará *relay*, esto es, no hará reenvío de correo para otros que se conecten de forma externa a la máquina. Así se evita que puedan usar nuestro servidor para el reenvío masivo de correo, publicidad, etc. (*spamming*). No obstante, podemos indicar que si reenviaremos correo para ciertos dominios. Esto se usa en caso de que nuestro servidor esté configurado como servidor secundario (de reserva) de algún otro dominio, para lo que, de nuevo, deberemos coordinar–nos con el servidor de nombres del dominio correspondiente. En nuestro caso elegimos la opción por defecto: "none".

- Ahora podemos elegir para qué dominios queremos actuar como servidor de correo electrónico, además de para nuestro propio dominio. Esto es útil si por ejemplo somos servidor de correo secundario de otro dominio. Si elegimos "mx", nuestro sistema aceptará correo para aquellos dominios para los que haya una entrada MX en el servidor de nombres que indique a nuestro servidor como servidor de correo para esos dominios. En el caso más simple podemos responder simplemente "none".

- Ahora podemos elegir para que máquinas o rango de máquinas queremos actuar como servidor de correo electrónico. Por ejemplo, si actuamos como servidor de correo electrónico para el dominio *lab.es*, cuyas direcciones IP son de la forma 150.215.x.x y 150.216.7.x, podemos indicar aquí estas subredes como "150.215.0.0/16" y "150.216.7.0/24". Los números 16 y 24 representan el número de unos (1's) que tiene la máscara de subred. También podemos indicar nombres con comodines, por ejemplo "*.lab.es". Si los usuarios siempre mandan correo con clientes que corren en el propio servidor, podemos indicar aquí "none".

- Ahora debemos elegir el nombre del usuario al que se redirige el correo del superusuario (*root*). Esta debe ser la cuenta personal que tenga el administrador del sistema, para que no sea necesario usar la cuenta de *root* para leer el correo del administrador. Es recomendable que se emplee una

cuenta local en la misma máquina. Esta opción se puede cambiar más adelante editando el fichero `/etc/aliases`.

- Si ya teníamos un fichero `/etc/aliases`, se nos preguntará si queremos conservarlo o instalar uno nuevo. Si instalamos uno nuevo, perderemos los cambios realizados manualmente, pero el fichero antiguo se conservará con otro nombre.
- La última pantalla nos muestra un resumen de los datos que hemos suministrados y podemos elegir entre repetir el cuestionario o aceptar la configuración elegida.

5.1.2 Control del *relay* por host de origen

Hemos visto que el proceso de configuración con `eximconfig` podemos indicar las subredes para las que hacemos de servidor de correo electrónico. Cuando el número de estas subredes es elevado y/o los clientes son muchos y dispersos, es mejor guardar la lista de los clientes para los que hacemos relay en un fichero. Para ello editamos el fichero `/etc/exim/exim.conf` y buscamos la línea donde se define el parámetro *host_accept_relay*. Añadimos a su valor el nombre de un fichero como en el ejemplo siguiente:

```
# The setting below allows your host to be used as a mail relay on-
ly by
# localhost: it locks out the use of your host as a mail relay by any
# other host. See the section of the manual entitled "Control of relaying"
# for more info.

# Valor original
# host_accept_relay = 127.0.0.1 : ::::1
# Nuevo valor
host_accept_relay = 127.0.0.1 : ::::1 : /etc/exim/host_accept_relay
```

Ahora creamos el fichero `/etc/exim/host_accept_relay` y escribimos allí la lista de máquinas y redes que pueden usarnos como servidor de correo electrónico. Por ejemplo:

```
150.215.0.0/16
150.216.7.0/24
*.lab.es
pc17.fac.es
150.214.141.10
```

Vemos que es posible indicar tanto direcciones IP concretas (150.214.141.10), subredes (150.216.7.0/24), subdominios (*.lab.es) y nombres de máquinas (pc17.fac.es). Cuando es posible es preferible el formato numérico pues así no dependemos del servidor de nombres.

5.1.3 Control del *relay* por dirección de correo de origen

Cuando los usuarios del correo electrónico se conectan siempre desde unas direcciones IP fijas, es posible darles acceso al servidor usando el método anterior, pero este método no se puede aplicar cuando los usuarios utilizan direcciones IP dinámicas (conexión por módem) o desean poder enviar correo usando cualquier máquina de Internet como cliente. En este caso, podemos autorizar el relay en función de la dirección de correo del usuario cliente.

Como en el apartado anterior, vamos a crear un fichero donde escribiremos las direcciones de los usuarios para los que se hace relay. Primero, incluimos las líneas siguientes en el fichero `/etc/exim/exim.conf` en un sitio apropiado, por ejemplo, tras la configuración de la opción *host_accept_relay*:

```
# Lista de direcciones origen para las que se hace relay.
relay_match_host_or_sender
sender_address_relay = /etc/exim/sender_address_relay
```

Ahora incluimos en el fichero `/etc/exim/sender_address_relay` las direcciones necesarias, por ejemplo:

```
# direcciones de paco
paco@frodo.lab.es
paco@lab.es
# direcciones de pepe
pepe@frodo.lab.es
pepe@lab.es
```

donde se han incluido las direcciones de los usuarios *paco* y *pepe*, indicando los diferentes nombres de dominio que pueden utilizar. Como puede verse, pueden incluirse líneas de comentario comenzando con #.

5.1.4 Control de `exim`

Control del método de inicio

Por defecto, `exim` se inicia bajo demanda controlado por `inetd`. Esta es la configuración adecuada para sistemas con poco tráfico de correo, pues evita tener el servidor corriendo constantemente y consumiendo recursos. Para sistemas con mucho tráfico de correo es mejor tener el servidor activo todo el tiempo. Para ello basta con comentar la línea que arranca el servicio *smtp* en el fichero `/etc/inetd.conf` con una simple “#” y ejecutar el script de iniciación independiente:

```
# /etc/init.d/exim start
```

El script detectará la configuración `/etc/inetd.conf` y ejecutará el servidor de forma permanente cada vez que se inicie la sesión.

Control del estado de `exim`

Con el programa gráfico `eximon(8)` que se encuentra en el paquete del mismo nombre, podemos ver el estado de las colas de mensajes de correo electrónico, los últimos errores generados y hacer operaciones sobre los mensajes pendientes de forma sencilla. El uso de este programa es de especial utilidad para saber por qué no se ha enviado aquel mensaje que enviamos hace dos días, además de otras manipulaciones posibles sobre los mensajes. Para realizar una acción sobre un mensaje se debe mantener pulsada la tecla *shift* mientras se pulsa el primer botón del ratón sobre el mensaje seleccionado.

5.1.5 Para saber más

`exim` es un programa muy complejo y altamente configurable y viene con una extensa documentación. Ésta viene en dos versiones: formato *info*, que se puede consultar desde dentro del editor `emacs` o desde el visor de ayuda del entorno *GNOME*, y en formato *html*. El primer formato se instala con el paquete `exim-doc` y el segundo con el paquete `exim-doc-html`.

5.2 Servidor de correo entrante: POP, IMAP

Para poder leer remotamente el correo electrónico que ha llegado a nuestro servidor de correo se emplean protocolos para recuperar correo como POP e IMAP. Estos protocolos están soportados por multitud de clientes como *Netscape Communicator* o *Microsoft Outlook*™.

5.2.1 Instalación de los servidores POP e IMAP

Existen varios servidores POP e IMAP incluidos con Debian. Unos de los más sencillos se encuentran en los paquetes `ipopd` e `uw-imapd`. Para ponerlos en marcha no se necesita configuración alguna, basta con instalar los paquetes.

El inicio de los servidores es controlado por `inetd`. El paquete `ipopd` incluye tanto el servidor de la versión 2 (POP2) como de la versión 3 (POP3). En la actualidad prácticamente no se utiliza el POP2, por lo que es recomendable desactivarlo comentando la línea correspondiente del fichero `/etc/inetd.conf`.

5.2.2 Conexión segura a POP e IMAP

En la conexión a los servidores POP e IMAP es necesario suministrar la clave del usuario para conectar al servicio. Ninguno de estos servidores emplea mecanismos de conexión segura, por lo que la conexión remota desde redes que no controlemos entraña graves riesgos de seguridad, pues cualquier máquina en las redes de origen o destino podría escanear la red y averiguar las claves suministradas. Para evitar esto es muy recomendable emplear conexiones seguras (mediante el protocolo SSL) para conectar a los servidores POP e IMAP. Existen dos alternativas (y probablemente más de dos) de hacer esto: instalar versiones con soporte SSL de los servidores POP e IMAP o instalar un paquete que haga de interfaz con los servidores inseguros tradicionales. Comentaremos ambas opciones.

Versiones seguras POP e IMAP

La instalación y configuración son tan fáciles como en el caso anterior. Basta con instalar los paquetes `ipopd-ssl` y `uw-imapd-ssl`, que sustituirán a las versiones inseguras si están instaladas. Estos paquetes también soportan conexiones inseguras, por lo que no es necesario tener instaladas ambas versiones. Los paquetes incluyen certificados autogenerados que se almacenan en `/etc/ssl/certs` y que pueden ser sustituidos por certificados *oficiales* o generados por nosotros.

Para mejorar la seguridad es muy recomendable deshabilitar la conexión insegura a los servicios, lo cual puede hacerse fácilmente comentando las líneas adecuadas en el fichero `/etc/inetd.conf`. Por ejemplo:

```
     #:MAIL: Mail, news and uucp services.
     smtp            stream  tcp     nowait  mail    /usr/sbin/exim ex-
im -bs
     # imap2  stream  tcp     nowait        root    /usr/sbin/tcpd  /usr/sbin/im
     # imap3  stream  tcp     nowait        root    /usr/sbin/tcpd  /usr/sbin/im
     imaps  stream  tcp     nowait        root    /usr/sbin/tcpd  /usr/sbin/imap
     # pop2     stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/ipop2d
     # pop3     stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/ipop3d
     pop3s    stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/ipop3d
```

Instalación y configuración de `sslwrap`

`sslwrap` hace de interfaz seguro mediante SSL para múltiples servicios, entre ellos POP e IMAP. Este programa actúa como un servidor a la espera de conexiones POP e IMAP seguras (servicios pop3s e imaps), hace las funciones de cifrado de la conexión y reenvía la conexión ya descifrada a los servidores reales. El paquete Debian se llama `sslwrap` y es fácil de configurar. `sslwrap` es muy flexible porque permite gestionar de forma centralizada la conexión segura a diversos servicios (no sólo POP e IMAP) y permite que los servicios se ejecuten en máquinas diferentes a donde funciona el programa `sslwrap`. A continuación veremos como configurar los servicios POP e IMAP de forma segura empleando este paquete.

Al instalar el paquete `sslwrap` entraremos en un asistente para su configuración controlado por el sistema de configuración de Debian *Debconf*. Los pasos a seguir son los siguientes:

- En la primera pantalla elegiremos activación por `inetd`.
- En la pantalla siguiente dejaremos la opción por defecto: “127.0.0.1”.
- En la pantalla siguiente diremos que SI queremos usar `sslwrap` con certificados. Esto es necesario para un correcto funcionamiento de los clientes.
- Si disponemos de un certificado propio creado con anterioridad, podemos indicar su localización aquí. En caso contrario dejamos el valor por defecto para que el certificado sea creado por `sslwrap`.

- El la siguiente ventana, si vamos a generar nuestro propio certificado, elegimos que SI para que sslwrap compruebe el certificado en el inicio y genere uno nuevo en cado de que éste haya caducado.
- Elegimos que SI a sobreescribir el certificado antiguo si sslwrap encuentra que estaba corrupto.
- En la siguiente ventana se nos mostrarán algunos consejos sobre el uso de `sslwrap` con algunos servicios
- Ahora podemos elegir qué servicios proteger con `sslwrap`. Elegiremos *imaps* y *pop3s*, que son la opción por defecto.

Caducidad y regeneración del certificado

Al iniciarse `sslwrap` por primera vez generará un nuevo fichero de certificado, si éste no existía. La primera vez que los clientes se conecten recibirán una advertencia ya que el certificado generado es *autofirmado* y no está avalado por una autoridad de certificación. Los clientes también recibirán una advertencia cuando caduque el certificado, que por defecto es un mes. Para regenerarlo basta con ejecutar el script de inicio de sslwrap:

```
# /etc/init.d/sslwrap start
```

Podemos hacer que este script se ejecute diariamente para que compruebe la validez del certificado si incluimos este comando en un script dentro del directorio `/etc/cron.daily`

Bloquear acceso directo a POP e IMAP

Una vez instalado `sslwrap` no debemos permitir el acceso directo a los servicios POP e IMAP, sino a las versiones seguras suministradas por `sslwrap`. Para ello podemos usar la siguiente configuración de los ficheros `hosts.deny` y `hosts.allow`, donde se supone una política de seguridad tipo *casi todo cerrado*.

```
# fichero /etc/hosts.deny
ALL: ALL

# fichero /etc/hosts.allow
ipop3d, imapd: localhost
sslwrap: ALL
```

Activar soporte SSL en los clientes

Para acceder a los servicios POP e IMAP en su versión segura es necesario configurar las conexiones POP e IMAP de los clientes para que usen conexión segura mediante SSL. La mayoría de los clientes actuales soportan esta opción, como *Netscape Communicator*, *mozilla*, *evolution* o *Microsoft Outlook*^{TM}. En cada caso se deberá consultar la documentación del cliente que se desee usar.

Capítulo 6

Servidor de ficheros SAMBA

Samba es un paquete de software de dominio público que permite compartir recursos por la red compatible con la red de Microsoft. Los principales servicios que ofrece son cuatro:

1. Servicios de archivos e impresoras.
2. Servicios de autentificación.
3. Servicios de resolución de nombres.
4. Servicios de listado (*browsing*).

Se puede encontrar la documentación en `http://www.samba.org`

6.1 Instalación

En la distribución debian, el paquete de instalación es *samba*, la instalación sería:

```
apt-get install samba
```

Durante el proceso de instalación se nos preguntará sobre el modo en el que se desea que se inicie el servicio SAMBA:

- Como demonio del sistema.
- Arranque bajo demanda *inetd*.[1]

En el caso de tener instalado el servicio bajo demanda, cuando realicemos cambios en la configuración será necesario reiniciar el servicio *inetd* de la forma:

[1] Ver documento de administración del sistema.

```
killall -HUP inetd
```

El tratamiento de los usuarios de este servicio se hace de manera diferente al de los usuarios de la máquina, es por ello que durante el proceso de instalación también se nos de la opción de generar los usuarios y contraseñas a partir de los usuarios y contraseñas existentes en la máquina.

6.2 Comandos de SAMBA

A continuación se muestra la relación de comandos que instala el paquete SAMBA así como una breve descripción de cada uno de ellos.

- *smbd:* Demonio del servidor SAMBA. Provee servicios de archivos e impresoras.
- *nmbd:* Demonio del servidor SAMBA. Provee servicios de nombres de *NetBios* y exploración del servidor
- *smbclient:* Implementa un cliente parecido al *ftp* pero para SAMBA.
- *testparm:* Es una simple utilidad para comprobar el fichero de configuración del servidor `smb.conf`
- *testprns:* Utilidad para comprobar la capacidad de impresión para las impresoras definidas en el fichero `printcap`
- *smbstatus:* Proporciona información sobre las conexiones actuales al cliente *smbd*
- *nmblookup:* Permite obtener los nombres *NetBios* a partir del fichero `hosts` de Unix
- *make_smbcodepage:* Utilidades para crear la pagina de código.
- *smbpasswd:* Permite cambiar las contraseñas del acceso de los usuarios al servicio y agregar nuevos usuarios.

6.3 Configuración de SAMBA

Para realizar la configuración de SAMBA basta con editar el fichero `/etc/samba/smb.conf`. En este fichero se pueden modificar todas las opciones que ofrece el paquete de software. Para realizar esta configuración también existen herramientas gráficas como *gnosamba*[2].

Existe también un programa llamado *sambaconfig* el cual permite reconfigurar SAMBA en cualquier momento, pudiéndose cambiar el modo de funcionamiento de demonio a inetd.

Tras la instalación nos encontramos una configuración por defecto, la cual permite a los usuarios de la maquina UNIX acceder a sus cuentas. Además instala el servidor de impresión. A continuación se muestra un ejemplo de configuración:

[2]En la distribución Debian el paquete es *gnosamba*

```
[global]
   printing = bsd
   printcap name = /etc/printcap
   load printers = yes
   guest account = nobody
   invalid users = root
   map to guest = bad password

; "security = user" is always a good idea. This will require a
; Unix account
;       in this server for every user accessing the server.
   security = user

; Change this for the workgroup your Samba server will part of
   workgroup = DTE

   server string = %h server (Samba %v)

; If you want Samba to log though syslog only then set the
; following
;       parameter to 'yes'. Please note that logging through
;  syslog in
;       Samba is still experimental.
   syslog only = no

; We want Samba to log a minimum amount of information to
; syslog. Everything
;       should go to /var/log/{smb,nmb} instead. If you
; want to log through
;       syslog you should set the following parameter to
; something higher.
   syslog = 0;

; This socket options really speed up Samba under Linux,
; according to my
;       own tests.
   socket options = IPTOS_LOWDELAY TCP_NODELAY SO_SNDBUF=4096
SO_RCVBUF=4096

; Passwords are encrypted by default. This way the latest
; Windows 95 and NT
;       clients can connect to the Samba server with no
; problems.
   encrypt passwords = yes
```

```
; It’s always a good idea to use a WINS server. If you
; want this server
;       to be the WINS server for your network change
; the following parameter
;       to "yes". Otherwise leave it as "no" and specify
; your WINS server
;       below (note: only one Samba server can be the
; WINS server).
;       Read BROWSING.txt for more details.
   wins support = no

; If this server is not the WINS server then specify who
; is it and uncomment
;       next line.
   wins server = 150.214.141.89

; Please read BROWSING.txt and set the next four
; parameters according
;       to your network setup. There is no valid default
; so they are commented
;       out.
   os level = 0
   domain master = no
   local master = no
   preferred master = no
; What naming service and in what order should we use
; to resolve host names
;       to IP addresses
   name resolve order = lmhosts host wins bcast

; This will prevent nmbd to search for NetBIOS names
;  through DNS.
   dns proxy = no

; Name mangling options

   preserve case = yes
   short preserve case = yes

; This boolean parameter controlls whether Samba attempts
;  to sync. the Unix
;       password with the SMB password when the encrypted
; SMB password in the
;       /etc/samba/smbpasswd file is changed.
   unix password sync = true
```

```
; For Unix password sync. to work on a Debian GNU/Linux
; system, the following
;       parameters must be set (thanks to Augustin Luton
;       (aluton@hybrigenics.fr) for sending the correct chat script for
;       the passwd program in Debian Potato).
   passwd program = /usr/bin/passwd %u
   passwd chat = *Enter\snew\sUNIX\spassword:* %n\n
*Retype\snew\sUNIX\spassword:* %n\n .

; The following parameter is useful only if you have the
; linpopup package
;       installed. The samba maintainer and the linpopup
; maintainer are
;       working to ease installation and configuration of
; linpopup and samba.
;    message command = /bin/sh -c '/usr/bin/linpopup "%f"
; "%m" %s; rm %s' &

; The default maximum log file size is 5 MBytes. That's
; too big so this
;       next parameter sets it to 1 MByte. Currently,
; Samba rotates log
;       files (/var/log/{smb,nmb} in Debian) when these
; files reach 1000 KBytes.
;       A better solution would be to have Samba rotate
; the log file upon
;       reception of a signal, but for now on, we have
; to live with this.
   max log size = 1000


[homes]
   comment = Home Directories
   browseable = no

; By default, the home directories are exported read only.
; Change next
;       parameter to "no" if you want to be able to write
;  to them.
   read only = yes

; File creation mask is set to 0700 for security reasons.
; If you want to
;       create files with group=rw permissions, set next
; parameter to 0775.
   create mask = 0700
```

```
; Directory creation mask is set to 0700 for security
;  reasons. If you want to
;       create dirs. with group=rw permissions, set
; next parameter to 0775.
   directory mask = 0700

[printers]
   comment = All Printers
   browseable = no
   path = /tmp
   printable = yes
   public = yes
   writable = no
   create mode = 0700

; A sample share for sharing your CD-ROM with others.
;[cdrom]
;   comment = Samba server's CD-ROM
;   writable = no
;   locking = no
;   path = /cdrom
;   public = yes
;
; The next two parameters show how to auto-mount a CD-ROM when the
;       cdrom share is accesed. For this to work /etc/fstab must contain
;       an entry like this:
;
; /dev/scd0   /cdrom  iso9660 defaults,noauto,ro,user   0 0
;
; The CD-ROM gets unmounted automatically after the
; connection to the
;
; If you don't want to use auto-mounting/unmounting
; make sure the CD
;       is mounted on /cdrom
;
;   preexec = /bin/mount /cdrom
;   postexec = /bin/umount /cdrom
```

En el fichero de configuración de SAMBA se definen los recursos mediante un nombre puesto entre corchetes. Cada uno de estos recursos aparecerá en la red de forma independiente, pudiendo tener cada uno de ellos su propia configuración. Esta configuración se realiza mediante directivas. Es destacable la existencia de un recurso denominado **[global]** el cual representa la configuración global del servidor. Bajo este recurso se pueden encontrar directivas tales como:

- Directivas de control de la impresión: *printing, printcap name, load printers*
- Directivas de control de acceso de los usuarios:*invalid users, guest account, map to guest, security ,hosts allow, etc.*
- Nombre del grupo de trabajo *workgroup*: permite definir el grupo de trabajo al que pertenecerá la maquina.
- Las opciones *wins* nos permiten definir quien es el servidor de nombres o crear un servidor de nombres en nuestra maquina
- Es posible mantener sincronizadas los cambios de contraseñas de los usuarios de la máquina UNIX mediante la directiva *unix password sync*

Para el resto de recursos que se van creando, el comportamiento se define de forma similar. El conjunto completo de las directivas que se pueden usar se encuentran en la documentación del servidor SAMBA.

6.4 Control de acceso

Los usuarios existentes en SAMBA se gestionan de forma diferente a los del sistema. Las contraseñas de estos usuarios pueden ser diferentes a las que poseen en sus cuentas. Es posible mantener sincronizadas las contraseñas mediante una directiva en la configuración global del servidor:

```
unix password sync = yes
```

También es posible gestionar estas contraseñas manualmente mediante un comando *smbpasswd*. Con este comando cada usuario puede cambiar su contraseñas de acceso a los recursos del servidor.

En el comando *smbpasswd* cabe destacar los siguientes argumentos en la línea de comandos:

- *–a:* Sirve para añadir un usuario a la lista de usuarios permitidos. Aunque existan los usuarios en el sistema LINUX, es necesario añadirlos de forma explícita para poder usarlos en SAMBA

  –d: Deshabilita el acceso mediante SAMBA a un usuario.

  –e: Habilita el acceso mediante SAMBA a un usuario.

  –x: Eliminar un usuario de la lista de acceso

6.5 Ejemplo de uso del servidor SAMBA

Como ejemplo se va a crear un directorio de acceso público situado en `/var/pub` de forma que cualquier usuario tenga acceso de lectura al mismo. Para ello basta con crear un nuevo recurso por ejemplo *[publico]*, ahora lo único que hay que hacer es poner las directivas concretas para que sea público, no tenga permiso de escritura y asociarlo a algún directorio de la máquina:

```
[publico]
   comment = Directorio publico
   writable = no
   locking = no
   path = /var/pub
   public = yes
   hosts allow = .dte.us.es 192.168.2.229
```

Es importante no olvidar que el directorio indicado */var/pub* contenga los permisos adecuados de lectura para que los usuarios los puedan leer. Por último se ha añadido la directiva *hosts allow*. Con esta directiva se limita el acceso al recurso a las máquinas indicadas, en esta caso solo tendrían acceso aquellas máquinas pertenecientes al dominio *dte* y la máquina con la IP 192.168.2.229.

6.6 Uso a nivel de usuario

Para usar SAMBA en primer lugar se debe conocer el mecanismo usado para identificar las diferentes maquinas en la red. Estas se identifican por un nombre, el cual se indica en el archivo de configuración del servidor. Estos nombres se preceden de doble barra. Por ejemplo *//pclab5* identifica a la maquina llamada *pclab5*. Si queremos acceder a un recurso concreto basta con escribir el nombre del recurso precedido de “/” tras el nombre de la maquina. Como ejemplo, la forma de identificar un directorio *publico* existente en *pclab5* seria *//pclab5/publico*.

Para el acceso a estos recursos SAMBA ofrece dos comandos, El primero es *smbclient* el cual implementa un cliente parecido al *ftp*.

```
$ smbclient //pclab5/pub -U pedro
```

en el ejemplo se muestra la forma de acceder al recurso *pub* en *//pclab5* identificándose como el usuario *pedro*. En el caso de no haber indicado el modificador –U, *smclient* se identificaría como el usuario que tiene la sesión iniciada actualmente en el terminal. Tras la correcta identificación aparece la linea de comando, con la cual se pueden ejecutar una serie de comandos para manipular los archivos. A continuación se muestra un ejemplo en el cual se pide ayuda a *smbclient* sobre los comandos existentes.

```
smb: \> ls
.               D        0  Mon Jun 25 20:02:30 2001
..              D        0  Mon Jun 25 20:02:30 2001

34537 blocks of size 65536. 27371 blocks available
smb: \> help
ls             dir            du             lcd            cd
pwd            get            mget           put            mput
rename         more           mask           del            open
rm             mkdir          md             rmdir          rd
prompt         recurse        translate      lowercase      print
```

```
printmode      queue          cancel         quit           q
exit           newer          archive        tar            blocksize
tarmode        setmode        help           ?              history
!
smb: \>
```

Otra forma de acceder a los ficheros compartidos en otras maquinas consiste en montar un recurso remoto en el sistema de ficheros de la maquina local. Esto es posible mediante el comando *sbmmount*.

```
$ mkdir prueba
$ smbmount //pclab5/publico prueba/
```

De esta forma se consigue tener en un directorio proprio llamado *prueba* montado un recurso compartido con el servidor SAMBA recurso de red. Este comando suele estar disponible para todos los ususarios de la máquina.

Capítulo 7

Servidor de Web Apache

Apache es un servidor Web. Este servidor permite responder a las peticiones *http* según la configuración indicada al servidor. Toda la documentación referente al servidor se puede encontrar en `http://www.apache.org`.

Apache es un servidor modular, es decir, posee únicamente una funcionalidad muy básica. La ampliación de esta funcionalidad se consigue cargando módulos de forma dinámica para cada uno de los servicios que se quiera ofrecer.

7.1 Instalación del servidor

Para instalar el servidor bajo la distribución Debian hay que instalar el paquete *apache*, bien desde el programa *dselect* o con *apt–get* como es habitual. En ambos casos surgirán dependencias con el paquete *apache–common* por lo que también se instalara.

```
$ apt-get install apache
```

Es recomendable instalar también la documentación disponible en el paquete de Debian llamado *apache–doc*.

7.2 Comandos de Apache

Tras la instalación del servidor, se instalan una serie de comandos a continuación descritos:

- *httpd:* Servidor de paginas Web.
- *apachectl:* Es una utilidad de control del servidor. Permite detener o iniciar el servidor, así como permite comprobar la sintaxis de los ficheros de configuración en busca de errores. También es posible comprobar el estado del servicio a través de este comando.

- *ab:* Es una utilidad que permite comprobar el rendimiento del servidor instalado. Principalmente es útil para ver las peticiones servidas por unidad de tiempo.
- *apxs:* Esta utilidad permite instalar módulos de extensión en el servidor de forma dinámica. Para poder hacer esto es necesario habilitar esta posibilidad en el momento en que se inicie mediante el comando *httpd –l*.
- *dbmmanage, htdigest, htpasswd:* Permite crear y modificar los archivos usados para guardar usuarios y contraseñas para la autentificación básica de usuarios *http*.
- *logresolve:* Permite hacer la resolución de nombres para las direcciones IP que se encuentran el los archivos de *log*.
- *rotatelogs:* Permite continuar los archivos de seguimiento (logs) en otros archivos, sin necesidad de reiniciar el servidor.
- *suexec:* Cambia el usuario que ejecuta procesos del servidor.

7.3 Configuración del servidor

La configuración del servidor se realiza mediante la edición de los ficheros que se encuentran en el directorio */etc/apache/*. Esta configuración se realiza mediante el uso de directivas que se colocan en estos archivos de configuración. Estos archivos son:

- *httpd.conf:* Es la configuración global del servidor. Contiene el conjunto de opciones comunes a todo el servicio *http*
- *access.conf:* Contiene las directivas globales de control de acceso a los diferentes puntos del Web.
- *srm.conf:* En este fichero se puede realizar la configuración de las diferentes partes del Web que tengamos en el servidor, así como de los diferentes webs que tengamos.
- *mime.types:* Contiene información sobre el modo en que el servidor debe interpretar los diferentes tipos de archivos existentes en el Web.

7.4 Directivas

Como se mencionó anteriormente los archivos de configuración del servidor están formadas por directivas. Existen una serie de reglas generales a la hora de incluir nuevas directivas en estos archivos de configuración. En primer lugar, solo se puede colocar un directiva por linea, teniendo la posibilidad de ocupar varias lineas mediante el uso del carácter "\" al final de la misma, de esta forma indicamos que la linea no termina. Para poner un comentario se utiliza el carácter "#" al principio del comentario.

En este punto es conveniente mencionar que tras la edición de los archivos de configuración del servidor es recomendable usar el comando

```
apachectl configtest
```

con este comando nos aseguramos que los archivos de configuración no contienen errores sintácticos. A continuación se muestra en contenido del archivo de configuración global del servidor *httpd.config*

```
# This is the main server configuration file. See URL
# http://www.apache.org/
# for instructions.

# Do NOT simply read the instructions in here without
# understanding
# what they do, if you are unsure consult the online docs.
# You have been
# warned.

# Originally by Rob McCool

# Shared Object Module Loading:
# To be able to use the functionality of a module
# which was built
# as a shared object you have to place corresponding
# ‘LoadModule’
# lines at this location so the directives contained
# in it are
# actually available _before_ they are used.
# Example:

# ServerType is either inetd, or standalone.

ServerType standalone

# If you are running from inetd, go to "ServerAdmin".

# Port: The port the standalone listens to. For ports
# < 1023, you will
# need httpd to be run as root initially.

Port 80

# HostnameLookups: Log the names of clients or
# just their IP numbers
#   e.g.   www.apache.org (on) or 204.62.129.132 (off)
# The default is off because it’d be overall
# better for the net if people
# had to knowingly turn this feature on.
```

```
HostnameLookups off

# If you wish httpd to run as a different user or
# group, you must run
# httpd as root initially and it will switch.

# User/Group: The name (or #number) of the user/group
#  to run httpd as.
#  On SCO (ODT 3) use User nouser and Group nogroup
#  On HPUX you may not be able to use shared memory
#  as nobody, and the
#  suggested workaround is to create a user www and
# use that user.

User www-data
Group www-data

# ServerAdmin: Your address, where problems with
# the server should be
# e-mailed.

ServerAdmin webmaster@cronos.dte.us.es

# ServerRoot: The directory the server’s config,
# error, and log files
# are kept in.
# NOTE!  If you intend to place this on a NFS
# (or otherwise network)
# mounted filesystem then please read the LockFile
# documentation,
# you will save yourself a lot of trouble.

ServerRoot /etc/apache

# BindAddress: You can support virtual hosts with
#  this option. This option
# is used to tell the server which IP address to
# listen to. It can either
# contain "*", an IP address, or a fully qualified
# Internet domain name.
# See also the VirtualHost directive.

BindAddress *
```

```
# The Debian package of Apache loads every feature
# as shared modules.
# Please keep this LoadModule: line here, it is needed for installation.
# LoadModule vhost_alias_module /usr/lib/apache/
1.3/mod_vhost_alias.so
# LoadModule env_module /usr/lib/apache/1.3
/mod_env.so
LoadModule config_log_module /usr/lib/apache
/1.3/mod_log_config.so
# LoadModule mime_magic_module /usr/lib/apache/
1.3/mod_mime_magic.so
LoadModule mime_module /usr/lib/apache/1.3/mod_mime.so
LoadModule negotiation_module /usr/lib/apache/
1.3/mod_negotiation.so
LoadModule status_module /usr/lib/apache/1.3/
mod_status.so
# LoadModule info_module /usr/lib/apache/1.3/
mod_info.so
# LoadModule includes_module /usr/lib/apache/1.3/
mod_include.so
LoadModule autoindex_module /usr/lib/apache/1.3/
mod_autoindex.so
LoadModule dir_module /usr/lib/apache/1.3/
mod_dir.so
LoadModule cgi_module /usr/lib/apache/1.3/
mod_cgi.so
# LoadModule asis_module /usr/lib/apache/1.3/
mod_asis.so
# LoadModule imap_module /usr/lib/apache/1.3/
mod_imap.so
# LoadModule action_module /usr/lib/apache/1.3/
mod_actions.so
# LoadModule speling_module /usr/lib/apache/1.3/
mod_speling.so
LoadModule userdir_module /usr/lib/apache/1.3/
mod_userdir.so
LoadModule alias_module /usr/lib/apache/1.3/
mod_alias.so
LoadModule rewrite_module /usr/lib/apache/1.3/
mod_rewrite.so
LoadModule access_module /usr/lib/apache/1.3/
mod_access.so
LoadModule auth_module /usr/lib/apache/1.3/
mod_auth.so
# LoadModule anon_auth_module /usr/lib/apache/
1.3/mod_auth_anon.so
```

```
# LoadModule dbm_auth_module /usr/lib/apache/
1.3/mod_auth_dbm.so
# LoadModule db_auth_module /usr/lib/apache/
1.3/mod_auth_db.so
# LoadModule proxy_module /usr/lib/apache/1.3/
libproxy.so
# LoadModule digest_module /usr/lib/apache/1.3/
mod_digest.so
# LoadModule cern_meta_module /usr/lib/apache/
1.3/mod_cern_meta.so
LoadModule expires_module /usr/lib/apache/
1.3/mod_expires.so
# LoadModule headers_module /usr/lib/apache/
1.3/mod_headers.so
# LoadModule usertrack_module /usr/lib/apache/
1.3/mod_usertrack.so
LoadModule unique_id_module /usr/lib/apache/
1.3/mod_unique_id.so
LoadModule setenvif_module /usr/lib/apache/
1.3/mod_setenvif.so
# LoadModule sys_auth_module /usr/lib/apache/
1.3/mod_auth_sys.so
# LoadModule put_module /usr/lib/apache/1.3/mod_put.so
# LoadModule throttle_module /usr/lib/apache/
1.3/mod_throttle.so
# LoadModule allowdev_module /usr/lib/apache/
1.3/mod_allowdev.so
# LoadModule auth_mysql_module /usr/lib/apache/
1.3/mod_auth_mysql.so
# LoadModule pgsql_auth_module /usr/lib/apache/
1.3/mod_auth_pgsql.so
# LoadModule eaccess_module /usr/lib/apache/
1.3/mod_eaccess.so
# LoadModule roaming_module /usr/lib/apache/
1.3/mod_roaming.so

ExtendedStatus on

# ErrorLog: The location of the error log file.
# If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog /var/log/apache/error.log

# LogLevel: Control the number of messages logged
# to the error_log.
```

```
# Possible values include: debug, info, notice,
# warn, error, crit,
# alert, emerg.

LogLevel warn

# The following directives define some format
# nicknames for use with
# a CustomLog directive (see below).

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"
%{User-Agent}i\" %T %v" full
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"
%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# The location of the access logfile (Common Logfile
# Format).
# If this does not start with /, ServerRoot is
# prepended to it.

CustomLog /var/log/apache/access.log common

# If you would like to have an agent and referer
# logfile uncomment the
# following directives.

#CustomLog /var/log/apache/referer.log referer
#CustomLog /var/log/apache/agent.log agent

# If you prefer a single logfile with access, agent
# and referer information
# (Combined Logfile Format) you can use the following
# directive.

#CustomLog /var/log/apache/access.log combined

# PidFile: The file the server should log its pid to
PidFile /var/run/apache.pid

# ScoreBoardFile: File used to store internal server
# process information.
# Not all architectures require this.  But if yours
# does (you'll know because
```

```
# this file is created when you run Apache) then you
# *must* ensure that
# no two invocations of Apache share the same
#  scoreboard file.
# ScoreBoardFile logs/apache_runtime_status

# The LockFile directive sets the path to the lockfile
# used when Apache
# is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or
# USE_FLOCK_SERIALIZED_ACCEPT. This directive should
# normally be left at
# its default value. The main reason for changing
# it is if the logs
# directory is NFS mounted, since the lockfile MUST
# BE STORED ON A LOCAL
# DISK. The PID of the main server process is
# automatically appended to
# the filename.
#
LockFile /var/run/apache.lock

# ServerName allows you to set a host name which is
# sent back to clients for
# your server if it’s different than the one the
# program would get (i.e. use
# "www" instead of the host’s real name).
#
# Note: You cannot just invent host names and hope
# they work. The name you
# define here must be a valid DNS name for your
# host. If you don’t understand
# this, ask your network administrator.

#ServerName new.host.name

# UseCanonicalName:  (new for 1.3)  With this
# setting turned on, whenever
# Apache needs to construct a self-referencing URL
# (a url that refers back
# to the server the response is coming from) it
# will use ServerName and
# Port to form a "canonical" name.  With this
# setting off, Apache will
# use the hostname:port that the client supplied,
# when possible.  This
# also affects SERVER_NAME and SERVER_PORT in CGIs.
```

```
UseCanonicalName on

# CacheNegotiatedDocs: By default, Apache sends
# Pragma: no-cache with each
# document that was negotiated on the basis of
# content. This asks proxy
# servers not to cache the document. Uncommenting
# the following line disables
# this behavior, and proxies will be allowed to
# cache the documents.

#CacheNegotiatedDocs

# Timeout: The number of seconds before receives
# and sends time out

Timeout 300

# KeepAlive: Whether or not to allow persistent
# connections (more than
# one request per connection).
# Set to "Off" to deactivate.

KeepAlive On

# MaxKeepAliveRequests: The maximum number of
# requests to allow
# during a persistent connection. Set to 0 to allow
# an unlimited amount.
# We reccomend you leave this number high, for
# maximum performance.

MaxKeepAliveRequests 100

# KeepAliveTimeout: Number of seconds to wait
# for the next request

KeepAliveTimeout 15

# Server-pool size regulation.  Rather than making
# you guess how many
# server processes you need, Apache dynamically
# adapts to the load it
# sees --- that is, it tries to maintain enough
# server processes to
# handle the current load, plus a few spare
```

```
# servers to handle transient
# load spikes (e.g., multiple simultaneous
# requests from a single
# Netscape browser).

# It does this by periodically checking how
# many servers are waiting
# for a request.  If there are fewer than
# MinSpareServers, it creates
# a new spare.  If there are more than
# MaxSpareServers, some of the
# spares die off.  These values are
# probably OK for most sites ---

MinSpareServers 5
MaxSpareServers 10

# Number of servers to start --- should be a
# reasonable ballpark figure.

StartServers 5

# Limit on total number of servers running,
# i.e., limit on the number
# of clients who can simultaneously connect
# --- if this limit is ever
# reached, clients will be LOCKED OUT, so it
# should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a
# runaway server from taking
# Unix with it as it spirals down...

MaxClients 150

# MaxRequestsPerChild: the number of
# requests each child process is
#  allowed to process before the child dies.
#  The child will exit so as to avoid problems
# after prolonged use when
#  Apache (and maybe the libraries it uses) leak.
# On most systems, this
#  isn’t really needed, but a few (such as Solaris)
# do have notable leaks
#  in the libraries.

MaxRequestsPerChild 30
```

```
# Listen: Allows you to bind Apache to specific
# IP addresses and/or
# ports, in addition to the default. See also
#  the VirtualHost command

#Listen 3000
#Listen 12.34.56.78:80

# VirtualHost: Allows the daemon to respond to
# requests for more than one
# server address, if your server machine is
#  configured to accept IP packets
# for multiple addresses. This can be
# accomplished with the ifconfig
# alias flag, or through kernel patches like VIF.

# Any httpd.conf or srm.conf directive may go
# into a VirtualHost command.
# See also the BindAddress entry.

#<VirtualHost host.some_domain.com>#ServerAdmin webmaster@host.some_domain.
#DocumentRoot /var/www/host.some_domain.com
#ServerName host.some_domain.com
#ErrorLog /var/log/apache/host.some_domain.com-error.log
#TransferLog /var/log/apache/
#host.some_domain.com-access.log
#</VirtualHost>
```

En este archivo de configuración global se pueden destacar las siguientes directivas:

- *Port 80:* Indica el puerto en el que se colocara en Web
- *ServerRoot /etc/apache:* Con esta directiva indicamos donde se encuentran los archivos de configuración y de *log* del servidor.
- *LoadModule:* Esta es la directiva con la que se consiguen cargar en Apache los módulos que se deseen, a medida de los servicios que se quieran ofrecer con el servidor.

Con esta configuración básica instalada por defecto el servidor debe funcionar sin problemas.

7.5 Mapeo de URL

La forma en que Apache mapea la URL que esta sirviendo se configura en el archivo */etc/apache/srm.conf*, el cual se muestra a continuación:

```
# With this document, you define the name space that users see
# of your http
# server.  This file also defines server settings which affect
# how requests are
# serviced, and how results should be formatted.

# See the tutorials at http://www.apache.org/ for
# more information.

# Originally by Rob McCool; Adapted for Apache


# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this
# directory, but
# symbolic links and aliases may be used to point to other
# locations.
DocumentRoot /var/www

# UserDir: The name of the directory which is appended
# onto a user’s home
# directory if a ~user request is recieved.

UserDir public_html

# DirectoryIndex: Name of the file or files to use as a
# pre-written HTML
# directory index.  Separate multiple entries with spaces.

DirectoryIndex index.html

# FancyIndexing is whether you want fancy directory
# indexing or standard

FancyIndexing on

# AddIcon tells the server which icon to show
# for different files or filename
# extensions

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
```

```
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif */core
AddIcon /icons/deb.gif .deb Debian

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

# DefaultIcon is which icon to show for files which do not
# have an icon
# explicitly set.

DefaultIcon /icons/unknown.gif

# AddDescription allows you to place a short description
# after a file in
# server-generated indexes.
# Format: AddDescription "description" filename

# ReadmeName is the name of the README file the server
# will look for by
# default. Format: ReadmeName name
#
# The server will first look for name.html, include it if
# found, and it will
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
```

```
ReadmeName README
HeaderName HEADER

# IndexIgnore is a set of filenames which directory
# indexing should ignore
# Format: IndexIgnore name1 name2...

IndexIgnore .??* *~ *# HEADER HEADER.html README
README.html RCS CVS

# AccessFileName: The name of the file to look for
# in each directory
# for access control information.

AccessFileName .htaccess

# DefaultType is the default MIME type for documents
# which the server
# cannot find the type of from filename extensions.

DefaultType text/plain

# AddEncoding allows you to have certain browsers
# (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers
# support this.

AddEncoding x-compress Z
AddEncoding x-gzip gz

# AddLanguage allows you to specify the language
# of a document. You can
# then use content negotiation to give a browser
# a file in a language
# it can understand.  Note that the suffix does
# not have to be the same
# as the language keyword --- those with documents
# in Polish (whose
# net-standard language code is pl) may wish to
# use "AddLanguage pl .po"
# to avoid the ambiguity with the common suffix
# for perl scripts.

AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
```

```
AddLanguage da .da
AddLanguage it .it
AddLanguage es .es
AddLanguage br .br
AddLanguage ja .ja
AddLanguage dk .dk
AddLanguage pl .pl
AddLanguage kr .kr

# LanguagePriority allows you to give precedence
# to some languages
# in case of a tie during content negotiation.
# Just list the languages in decreasing order of
# preference.

LanguagePriority en fr de

# Default charset preference (see
# http://www.apache.org/info/css-security/).

AddDefaultCharset on
AddDefaultCharsetName iso-8859-1

# Redirect allows you to tell clients about
# documents which used to exist in
# your server’s namespace, but do not anymore.
# This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect fakename url


# Aliases: Add here as many aliases as you need
# (with no limit). The format is
# Alias fakename realname

# Note that if you include a trailing / on fakename
# then the server will
# require it to be present in the URL.  So "/icons"
# isn’t aliased in this
# example.

Alias /icons/ /usr/share/apache/icons/

# ScriptAlias: This controls which directories
# contain server scripts.
# Format: ScriptAlias fakename realname
```

```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

# If you want to use server side includes,
# or CGI outside
# ScriptAliased directories, uncomment the
# following lines.

# AddType allows you to tweak mime.types without
# actually editing it, or to
# make certain files to be certain types.
# Format: AddType type/subtype ext1

# For example, the PHP3 module (a separate
# Debian package)
# will typically use:
#AddType application/x-httpd-php3 .phtml
#AddType application/x-httpd-php3-source .phps

# AddHandler allows you to map certain file
# extensions to "handlers",
# actions unrelated to filetype. These can be
# either built into the server
# or added with the Action command (see below)
# Format: AddHandler action-name ext1

# To use CGI scripts:
#AddHandler cgi-script .cgi

# To use server-parsed HTML files
#AddType text/html .shtml
#AddHandler server-parsed .shtml

# Uncomment the following line to enable
# Apache’s send-asis HTTP file
# feature
#AddHandler send-as-is asis

# If you wish to use server-parsed imagemap files, use
#AddHandler imap-file map

# To enable type maps, you might want to use
#AddHandler type-map var

# Action lets you define media types that will
# execute a script whenever
```

```
# a matching file is called. This eliminates
# the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location

# Customizable error response (Apache style)
#  these come in three flavors
#
#    1) plain text
#ErrorDocument 500 "The server made a boo boo.
#  n.b.  the (") marks it as text, it does not get output
#
#    2) local redirects
#ErrorDocument 404 /missing.html
#  to redirect to local url /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#  n.b. can redirect to a script or a document
# using server-side-includes.
#
#    3) external redirects
#ErrorDocument 402 http://some.other_server.com
#/subscription_info.html
#

# mod_mime_magic allows the server to use various
# hints from the file itself
# to determine its type.
#MimeMagicFile conf/magic

# The following directives disable keepalives
# and HTTP header flushes.
# The first directive disables it for
# Netscape 2.x and browsers which
# spoof it. There are known problems with these.
# The second directive is for Microsoft Internet
# Explorer 4.0b2
# which has a broken HTTP/1.1 implementation
# and does not properly
# support keepalive when it is used on 301 or
# 302 (redirect) responses.

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 \
force-response-1.0
```

```
# The following directive disables HTTP/1.1
# responses to browsers which
# are in violation of the HTTP/1.0 spec by
# not being able to grok a
# basic 1.1 response.

BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

Alias /doc/ /usr/doc/
## The above line is for Debian Policy 3.0.1
#(FHS), which specifies that /doc
## is /usr/share/doc.  Packages should
#symlink to share/doc. -- apacheconfig
```

En este archivo es posible indicar la ubicación en nuestro sistema de ficheros de la URL con la directiva **DocumentRoot /var/www**. Con la directiva **DirectoryIndex index.html** indicamos cual es el documento por defecto que se debe abrir cuando se acceda a una URL sin indicar el archivo. Las directivas *AddIcon* permiten indicar los iconos que se deben mostrar en el caso de en el que se realiza la exploración de los archivos de un directorio vía Web. También es posible indicar cual es la página que se debe mostrar en caso de error en el servidor mediante la directiva **ErrorDocument**.

7.6 Ámbitos

El servidor Apache permite servir diferentes partes del Web cada una de configurada de forma diferente. Para ello usa un mecanismo que permite definir ámbitos, en los cuales se pueden colocar directivas, siendo estas directivas solo aplicable al ámbito en el que esta incluida.

La declaración de un ámbito se hace mediante dos directivas, una de apertura de ámbito y otra de cierre:

```
<Directory /usr/lib/cgi-bin>
AllowOverride None
Options ExecCGI FollowSymLinks
</Directory>
```

en el ejemplo se abre un ámbito con *<Directory /usr/lib/cgi–bin>* en el cual se colocan una serie de directivas aplicables en este caso a un directorio concreto. Tras esto se cierra el ámbito con *</Directory>*.

Un ejemplo del uso de estos ámbitos o contextos se encuentra en el archivo */apache/access.conf* el cual se muestra a continuación.

```
# access.conf: Global access configuration
# Online docs at http://www.apache.org/
```

```
# This file defines server settings which affect
# which types of services
# are allowed, and in what circumstances.

# Each directory to which Apache has access, can be
# configured with respect
# to which services and features are allowed and/or
# disabled in that
# directory (and its subdirectories).

# Originally by Rob McCool

# This should be changed to whatever you set
# DocumentRoot to.

<Directory /var/www>

# This may also be "None", "All", or any
# combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or
# "MultiViews".

# Note that "MultiViews" must be named
# *explicitly* --- "Options All"
# doesn't give it to you (or at least, not yet).

Options Indexes FollowSymLinks

# This controls which options the .htaccess
# files in directories can
# override. Can also be "All", or any combination
# of "Options", "FileInfo",
# "AuthConfig", and "Limit"

AllowOverride None

# Controls who can get stuff from this server.

order allow,deny
allow from all

</Directory>

# /usr/lib/cgi-bin should be changed to
# whatever your ScriptAliased
```

```
# CGI directory exists, if you have that configured.

<Directory /usr/lib/cgi-bin>
AllowOverride None
Options ExecCGI FollowSymLinks
</Directory>

# Allow server status reports, with the URL of
# http://servername/server-status
# Change the ".your_domain.com" to match your
# domain to enable.

#<Location /server-status>
#SetHandler server-status

#order deny,allow
#deny from all
#allow from .your_domain.com
#</Location>

# Allow server info reports, with the URL of
# http://servername/server-info
# Change the ".your_domain.com" to match
# your domain to enable.

#<Location /server-info>
#SetHandler server-info

#order deny,allow
#deny from all
#allow from .your_domain.com
#</Location>

# Debian Policy assumes /usr/doc is "/doc/",
# at least from the localhost.

<Directory /usr/doc>
Options Indexes FollowSymLinks
AllowOverride None
order deny,allow
deny from all
allow from localhost
</Directory>

# This sets the viewable location of the
# mod_throttle status display.
```

```
#
# <location /throttle-info>
# SetHandler throttle-info
# </location>

# Do not allow users to browse foreign files
# using symlinks in
# their private webspace public_html.
# Note: This should be changed if you modify
# the UserDir-Option.
# We would really like to use LocationMatch
# but the Option we want
# is ignored with that directive.

<DirectoryMatch ^/home/.*/public_html>
Options Indexes SymLinksIfOwnerMatch
AllowOverride None
</DirectoryMatch>

# Do not allow retrieval of the override files,
# a standard security measure.
<Files .htaccess>
order allow,deny
deny from all
</Files>

# You may place any other directories or locations you wish to have
# access information for after this one.
```

Con estas definiciones que puede conseguir un concreto comportamiento del Web, por ejemplo, es posible denegar el acceso a ciertos archivos del mismo modo que se hace en el ejemplo anterior:

```
<Files .htaccess>
order allow,deny
deny from all
</Files>
```

con esto se consigue que ningún archivo *.htaccess* pueda ser leído o descargado por http.

7.7 Seguridad

Apache posee una serie de directivas que permiten limitar los accesos al servidor de diferentes maneras. Todas ellas pueden ser colocadas a lo largo del archivo de configuración, o en un ámbito determinado, de forma que solo se aplique a una zona del Web. Es destacable la directiva *BindAddress* la cual limita las maquinas que pueden acceder al servidor:

```
BindAddress *.dte.us.es
```

esta directiva limitaría el acceso al servidor al dominio *.dte.us.es*.

Otra forma de controlar el acceso a partes del Web se indicó en el apartado anterior, mediante el uso de ámbitos de directorio y archivo e indicando en estos ámbitos directivas de acceso. Por ultimo, mencionar la existencia de unos archivos especiales llamados *.htaccess*. Estos archivos permiten descentralizar la administración del Web y se pueden colocar a lo largo del sistema de archivos. Estos archivos están formados por directivas que se aplican sólo al directorio en el cual se encuentren y a los subdirectorio que partan de él. Con esto también se consigue que un usuario pueda administrar de manera independiente una parte del un Web y además pueda controlar el comportamiento de la parte que administra.

7.8 Ejemplo de uso

Como ejemplo de uso de Apache se van a crear dos webs dentro de la misma máquina. La forma que tendrá el usuario de acceder a una u otra, será indicando en el URL el puerto de la máquina al que quiere conectarse. Situaremos el primer sitio Web en `http://www.miservidor.com` siendo este el Web por defecto, y en el puerto 8080 colocaremos otro, de forma que desde un explorador de Internet habría que acceder con la URL `http://www.miservidor.com:8080`.

Para hacer esto en primer lugar se deben crear dos directorios diferentes en los que se albergarán los dos sitios Web. Para este ejemplo se han creado los directorios llamados */var/www/web1* y */var/www/web2*. Para conseguir esto es necesario incluir en el fichero *httpd.conf* las siguientes directivas:

```
Listen 80
Listen 8080
ServerName www.miservidor.com
DocumentRoot /var/www/web1

<VirtualHost 192.168.1.3:8080>
DocumentRoot /var/www/web2
</VirtualHost>
```

En el ejemplo se puede observar que es indiferente el uso de nombres como puede ser `www.miservidor.com` o directamente la dirección IP.

7.9 Características avanzadas

Apache posee una serie de características avanzadas para la administración de servidores Web que requieran funcionalidades complejas. Todas ellas se pueden consultar en la documentación de Apache. A continuación se van a enumerar solo desde un nivel descriptivo aquellas que se consideran mas interesantes:

- *Virtual Hosting:* Es una de las partes mas potentes de Apache, la cual permite tener e un mismo servidor varios sitios Web, es capaz de mostrar uno u otro según se haga el acceso de una determinada forma: por nombre, por puerto, por dirección ip, desde determinados orígenes, etc.
- *Creación de subdominios:* Es posible la creación de subdominios en el servidor. Por ejemplo, para el dominio `www.miservidor.com` podemos añadir nombres de la forma `www.otroservicio.miservidor.com`
- *Redirección:* Se puede hacer la redirección de peticiones al servidor. Por ejemplo:

  ```
  Redirect /soporte http://www.soporte.com
  ```

  con esto conseguiríamos que una petición al servidor de la forma `http://www.miservidor.com/soporte` se redirija a la dirección `http://www.soporte.com`.

Capítulo 8

Servidor Proxy SQUID

SQUID es un proxy para los servicios *ftp, http, https*. Toda la documentación sobre el este servidor proxy la podemos encontrar en las siguientes direcciones de Internet:

- `http://www.squid-cache.org`
- `http://squid.nlanr.net`
- `http://cache.is.co.za/squid`

Haciendo un pequeño resumen, inicialmente los servidores proxy se utilizaban como cachés de Internet. Actualmente ofrecen gran cantidad de servicios. El uso mas habitual consiste en colocarlo en un ordenador que tenga dos conexiones de red, una hacia Internet y otra hacia una red local, de forma que a través del proxy se consigue dar acceso a Internet a toda la red local.

8.1 Instalación

El paquete de instalación en la distribución de Linux Debian es *squid* la instalación sería:

```
apt-get install squid
```

Tras la instalación se instala el servicio correspondiente al servidor proxy SQUID en `/etc/init.d/squid`. La configuración del servidor queda centralizada en el fichero `/etc/squid.conf`

8.2 Configuración del servidor

En el archivo de configuración */etc/squid.conf* hay una serie de directivas que definen el comportamiento del servidor proxy.

Si editamos el archivo de configuración tras la instalación se puede observar que todas las directivas de control de los servicios están deshabilitadas. Por omisión, SQUID funciona con unos puertos predeterminados para cada uno de sus servicios. Esto puede ser problemático en muchos casos, ya que puede haber algún otro servicio funcionando en el puerto por defecto. Un caso en el que sucede esto es cuando tenemos en la misma máquina el servidor Apache y Squid. Ambos para se colocan en el puerto 80 por defecto, de forma que si otra máquina usa el proxy en el puerto 80 el resultado es que siempre responde con la página que tenga por defecto Apache en el puerto 80 no pudiendo acceder a ningún otro sitio de Internet.

Existen directivas que nos permiten colocar los diferentes servicios en diferentes puertos, únicamente hay que ir descomentándolos. En el caso del servicio *http*, se puede colocar en otro puerto de la siguiente forma:

```
http_port 3128
```

con esto la colocamos en el puerto 3128.

Tras reiniciar el servicio tenemos en el puerto 3128 el proxy funcionando para ofrecer servicios *http*. Otro problema es que en la configuración por defecto, solo se permite el uso local, es decir, ningún ordenador de la red lo puede usar nuestra máquina como servidor proxy. Para habilitar el acceso a otras máquinas es necesario manipular las listas de acceso, las cuales se explican en la siguiente sección.

8.3 Control de acceso

SQUID tiene un complejo sistema de control de acceso. Este es una de las características más difícil de utilizar en el servidor, si se consigue una buena configuración podría llegarse a usar como firewall.

Este sistema de control de acceso se basa en las denominadas **listas de control de acceso (ACL)**. Estas listas de control de acceso se declaran en el archivo de configuración mediante la directiva *acl*. A continuación se muestra un ejemplo de una lista de acceso:

```
acl dte srcdomain   dte.us.es
```

Esta lista de acceso contiene todos los clientes que se conecten desde el dominio *dte.us.es*. Cuando ya tenemos declarada una lista ACL, se puede utilizar para permitir o denegar el acceso al proxy . Por ejemplo si queremos permitir el acceso a la lista ACL *dte* escribiríamos la siguiente directiva en el archivo de configuración de SQUID:

```
http_access allow dte
```

Cada uno de los servicios de SQUID se configura con su propio conjunto de directivas. En este caso se ha habilitado el acceso al servicio *http* al dominio *dte.us.es*

De forma general, la sintaxis la directiva *acl* es de la siguiente forma: *“acl aclname acltype string”* donde *acltname* es el nombre que se le da a la lista y *acltype* indica el tipo de lista que se puede hacer. Los diferentes tipos serían:

- *src:* Mira la IP del cliente. Para crear una lista que contenga toda la red 192.168.1.0 escribiríamos

  ```
  acl laboratorio 192.168.1.0/24
  ```

  Si quisiéramos usar una sola IP o varias, bastaría con escribirlas seguidas.

  ```
  acl laboratorio 192.168.1.4 192.168.1.6
  ```

- *dst:* Crearía una lista en la que estarían las direcciones IP indicadas pero como destino. Es útil para poder denegar accesos a determinadas direcciones IP.

- *srcdomain:* Squid puede averiguar el dominio del cliente haciendo lo que se denomina DNS inverso. Con esto podemos crear listas de acceso que identifiquen a los clientes del servidor con un determinado dominio.

  ```
  acl dte srcdomain dte.us.es
  ```

  La lista *dte* identifica a todos los clientes que se conectan al proxy desde el dominio *dte.us.es*.

- *dstdomain:* Identifica los dominios destino hacia los cuales intentan acceder los clientes. Son útiles para denegar o permitir ciertos dominios completos.

  ```
  acl terra dstdomain terra.es
  http_access deny terra
  ```

  Con esto conseguimos que nadie pueda conectarse a los dominios de terra

- *srcdom_regex, dstdom_regex:* Funcionan de forma similar a *srcdomain* y *dstdomain* sólo que en vez de identificar a un dominio completo, lo que hace es buscar una expresión determinada dentro del la URL a la que se está accediendo. Si quisiéramos denegar el acceso a todas aquellas direcciones que contuvieran la palabra *sex* habría que incluir lo siguiente:

  ```
  acl sex dstdom_regex sex
  http_access deny sex
  ```

- *time:* Permite crear listas para poder poner restricciones de hora.

  ```
  acl ACLTIME time M 9:00-17:00
  ```

  Esta lista referencia los lunes de 9 a 17.

- *port:* Con esto conseguimos controlar los accesos a los diferentes puertos.

  ```
  acl puertohttp port 80
  http_access deny puertohttp
  ```

Así denegamos el acceso al puerto 80

Existen muchas más posibilidades las cuales se pueden consultar en el manual de referencia de SQUID.

La configuración que trae por defecto el servidor incluye algunas listas ya creadas y algunos accesos permitidos y denegados:

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 80 21 443 563 70 210 1025-65535
acl purge method PURGE
acl CONNECT method CONNECT
...
http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
...
http_access deny all
```

Es importante el orden en el que se coloquen los controles de acceso ya que si en primer lugar colocamos

```
http_access deny all
```

el resto de las directivas *allow* ya no tendrían efecto ya que todas estas directivas se procesan en orden consecutivo.

8.4 Configuraciones avanzadas

En esta sección se van a comentar algunas de las posibilidades que ofrece SQUID, éstas se pueden concretar en la propia documentación el servidor. Entre las múltiples opciones caben destacar:

- Posibilidad de limitar los recursos a utilizar en la máquina: memoria, espacio en disco procesos, etc.
- Posibilidad de configurar la política de reemplazo de los datos almacenados en el caché.
- SQUID está preparado para usarse conjuntamente con otras máquinas también con SQUID instalado, de forma que se puede conseguir una jerarquía de cachés en la red.
- También es posible usar SQUID para balancear la carga de un sitio WEB. El servidor es capaz de redirigir peticiones a diferentes máquinas para no colapsar ningún servicio en una red de servidores.

Capítulo 9

Servidor FTP

El servidor ftp que se va a usar es el llamado *proftd*. En la distribución Debian el paquete es *proftpd*. La documentación referente a este servidor la podemos encontrar en `http://www.proftpd.net`.

El diseño de este servidor es muy parecido al servidor Apache en conceptos como diseño modular y formato de los archivos de configuración, ámbitos, etc.

9.1 Instalación

La instalación consistiría en la ejecución del comando:

```
apt-get install proftpd
```

Durante el proceso de instalación se nos pregunta si deseamos habilitar el acceso mediante ftp anónimo, contestaremos sí, y con esto se nos creará un usuario llamado *ftp*.

Tras la instalación el servicio correspondiente al servidor ftp estará en el directorio `/etc/init.d/` es con el nombre de `proftpd`.

9.2 Configuración

La configuración del servidor se realiza mediante la edición del archivo `/etc/proftpd.conf`

A continuación se muestra el contenido del mismo:

```
# This is a basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use.  It establishes a single server
# and a single anonymous login.  It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.
```

```
ServerName "Debian"
ServerType standalone
DeferWelcome off

ShowSymlinks on
MultilineRFC2228 on
DefaultServer on
ShowSymlinks on
AllowOverwrite on

TimeoutNoTransfer 600
TimeoutStalled 600
TimeoutIdle 1200

DisplayLogin                   welcome.msg
DisplayFirstChdir              .message
LsDefaultOptions               "-l"

# Port 21 is the standard FTP port.
Port 21

# Umask 022 is a good standard umask to prevent new
# files and dirs
# (second parm) from being group and world writable.
Umask 022  022

# To prevent DoS attacks, set the maximum number
#  of child processes
# to 30.  If you need to allow more than 30
#  concurrent connections
# at once, simply increase this value.  Note
# that this ONLY works
# in standalone mode, in inetd mode you should
# use an inetd server
# that allows you to limit maximum number of
#  processes per service
# (such as xinetd)
MaxInstances 30

# Set the user and group that the server
# normally runs at.
User nobody
Group nogroup

# Normally, we want files to be overwriteable.
<Directory /*>
```

```
  AllowOverwrite on
</Directory>

# A basic anonymous configuration, no upload
# directories.

<Anonymous ~ftp>
  User ftp
  Group nogroup
  # We want clients to be able to login with
# "anonymous" as well as "ftp"
  UserAlias anonymous ftp

  RequireValidShell off

  # Limit the maximum number of anonymous logins
  MaxClients 10

  # We want ’welcome.msg’ displayed at login,
#  and ’.message’ displayed
  # in each newly chdired directory.
  DisplayLogin welcome.msg
  DisplayFirstChdir .message

  # Limit WRITE everywhere in the anonymous
#  chroot
  <Directory *>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>

  # Uncomment this if you’re brave.
  # <Directory incoming>
  #             <Limit READ WRITE>
  #             DenyAll
  #             </Limit>
  #             <Limit STOR>
  #             AllowAll
  #             </Limit>
  # </Directory>

</Anonymous>
```

Entre las primeras directivas que se encuentran se pueden configurar parámetros globales como son el nombre del servidor, el archivo que contiene el mensaje de bienvenida al servidor y el puerto en el que

se va a colocar.

Cuando se está utilizando el servidor ftp anónimo es importante revisar la directiva *umask*. Esta directiva controla los permisos con los que se crearán los nuevos archivos que se suban al servidor. Por defecto tenemos la siguiente configuración:

```
Umask  022  022
```

El primer parámetro es la máscara de permisos para los archivos y el segundo corresponde a los directorios. Con esto se consigue que se tenga sólo permiso de lectura sobre los nuevos archivos creados.

Otro detalle importante que se observa en la configuración es la posibilidad de crear *ámbitos* de manera similar a como se hacía en el servidor Apache:

```
<Directory /*>
AllowOverwrite on
</Directory>
```

Recordemos que esto nos permitía colocar directivas que sólo se aplican al ámbito en el que están incluidas. De hecho en el archivo de configuración se puede observar que el control del usuario anónimo se realiza mediante la declaración de un ámbito :

```
<Anonymous ~ftp>
...
  <Directory *>
    <Limit WRITE>
        DenyAll
    </Limit>
  </Directory>
...
</Anonymous>
```

incluso en un ámbito es posible anidar otro, como se hace en el ejemplo anterior. Se consigue poner restricciones a directorios solamente para el usuario anónimo.

9.3 Ejemplo con servidor anónimo

Por defecto cuando se habilita durante el proceso de instalación el acceso anónimo al servidor ftp, se crea un usuario llamado *ftp*, para el cual se crea una cuenta en el directorio `/home/ftp`. Es posible habilitar la escritura de este usuario de manera que se pueden subir archivos al servidor anónimo incluyendo en el archivo de configuración lo siguiente:

```
<Anonymous ~ftp>
...
```

```
    <Directory *>
      <Limit WRITE>
          AllowAll
      </Limit>
    </Directory>
...
</Anonymous>
```

Pero esto tiene un problema, todos los archivos que se están subiendo al servidor quedan ubicados en el directorio `/home/ftp` y esto no es conveniente. Lo ideal sería ubicar un directorio *ftp* en */var/* con los permisos adecuados, para ello desde la cuenta de root hacemos lo siguiente:

```
mkdir /var/ftp
chown ftp var/ftp/
```

por último para indicar el directorio al que accede el usuario anónimo basta con incluirlo en la cabecera donde se define el ámbito del usuario anónimo:

```
<Anonymous /var/ftp>
...
</Anonymous>
```

Capítulo 10

Servicio de Nombre de Dominio (DNS)

En el capítulo 'Introducción a Internet' en la página 3 hablamos brevemente de los nombres de dominio y como permiten identificar a máquinas en Internet mediante un nombre como *teclix.dte.us.es* en vez de la correspondiente dirección IP. Los ordenadores encargados de hacer esta traducción de nombres en direcciones IP son los Servidores de Nombres de Dominio o DNS. En este capítulo veremos como se instala y configura un servidor DNS en Debian. Este capítulo no es una descripción exhaustiva de los DNS, para ello existe documentación mucho más adecuada, como el *DNS–HOWTO*, que se instala junto con otros documentos similares con el paquete `doc-linux-html`, o su traducción al castellano, no tan actualizada, instalada con el paquete `doc-linux-es`. En estos documentos puede encontrar así mismo enlaces a la versión más reciente en Internet.

Por nuestra parte, nos centraremos en dos ejemplos: la configuración de un DNS *caché* y la de un DNS para un dominio hipotético que llamaremos *.linux.aula*.

10.1 Conceptos fundamentales de DNS

Un servidor DNS puede ser "responsable" de uno o más dominios. Por ejemplo, el DNS responsable (o autorizado en términos de DNS) del dominio *.linux.org* posee la información necesaria para traducir los nombres de hosts que terminen en *.linux.org*, como *www.linux.org*.

A un servidor DNS podemos preguntarle cualquier nombre. El DNS entonces comprobará si ese nombre está en su lista de nombres más solicitados últimamente (*caché*) en cuyo caso responderá inmediatamente con la dirección IP correspondiente. También pude ocurrir que el nombre solicitado sea de un dominio para el que este servidor está autorizado. Entonces, en la mayoría de los casos, el servidor sólo tendrá que buscar la dirección IP correspondiente en su tabla local, respondiendo directamente. En muchos casos, el servidor de nombre no conocerá la dirección IP correspondiente, en cuyo caso preguntará a alguno de los servidores de nombres generales que están distribuidos por Internet. Éstos se denominan servidores raíz (*root servers*). Estos servidores se encargarán de buscar la información solicitada preguntando a otros servidores de nombres si es necesario, y generarán una respuesta, que el primer DNS devolverá a su vez. De esta forma, el servicio de nombre se organiza como un árbol que parte de los servidores raíz y se extiende hacia los servidores de los distintos dominios y subdominios.

Los DNS no sólo traducen nombres a direcciones IP, también pueden traducir direcciones IP a sus nombres correspondientes haciendo lo que se denomina traducción inversa. El dominio especial *.in–addr.arpa* se emplea para realizar esta traducción, como veremos en los ejemplo. Además, los servidores de nombres suministran otros tipos de información relacionada con los nombres, como por ejemplo:

- Nombre del DNS responsable de un determinado dominio.
- Nombres de servidores de correo electrónico correspondientes a un dominio, pudiendo existir varios con distinta prioridad.
- Nombres alternativos correspondientes a un nombre principal, por ejemplo para hacer que tanto las direcciones *www.dte.us.es* como *teclix.dte.us.es* correspondan a la misma máquina.

10.2 Instalación del servicio de nombres en Debian

El servidor de nombres más utilizado con amplia diferencia en Linux y en la mayoría de los sistemas UNIX es BIND. En Debian la versión 9 se instala con el paquete `bind9` y su documentación con el paquete `bind9-doc`. Por tanto, basta con que hagamos:

```
# apt-get install bind9 bind9-doc
```

para tener instalado nuestro flamante servidor de nombres.

10.3 Configuración de un DNS–Caché

La operación más sencilla de un DNS es la actuación como simple *caché* o intermediario hacia otros servidores de nombres, sin ser responsable de ningún dominio en concreto. Al pedir un nombre a nuestro DNS éste simplemente preguntará a otros servidores de nombre, pero almacenará las últimas respuestas, acelerando considerablemente posteriores consultas a las mismas direcciones.

En Debian, bind viene configurado *de serie* para funcionar de esta forma, así que lo único que queda por hacer para beneficiarnos de nuestro propio servidor de nombres es configurar el sistema para que le pregunte a nuestro propio DNS en vez de a un DNS externo, mediante la configuración adecuada del fichero `/etc/resolv.conf`:

```
search dte.us.es us.es
nameserver 127.0.0.1
```

donde simplemente se ha indicado la dirección IP 127.0.0.1, que siempre corresponde a la propia máquina, como dirección del servidor de nombres. Si disponemos de conexión a Internet, podemos comprobar la configuración con los comandos `host(1)` o `dig(1)`:

```
$ host www.debian.org
www.debian.org       A 198.186.203.20

$ dig www.debian.org
; <<>> DiG 9.2.1 <<>> www.debian.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62570
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 8, ADDITIONAL: 5

;; QUESTION SECTION:
;www.debian.org. IN A

;; ANSWER SECTION:
www.debian.org. 10795 IN A 198.186.203.20
...
```

Por defecto, nuestro DNS caché preguntará a alguno de los servidores raíz cuando no conozca un nombre, pero es una buena idea configurarlo para que pregunte a un DNS más próximo, por ejemplo a uno de nuestra propia organización, universidad, etc. Podemos indicar uno o más de estos servidores en la sección *forwarder* del fichero de configuración de bind `/etc/bind/named.conf`. Por ejemplo, la primera parte del fichero de configuración por defecto con esta modificación queda:

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//

options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you might need to uncomment the query-source
        // directive below.  Previous versions of BIND always asked
        // questions using port 53, but BIND 8.1 and later use an unprivile
        // port by default.

        // query-source address * port 53;

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the address-
es replacing
```

```
                // the all-0's placeholder.

                // forwarders {
                //      0.0.0.0;
                // };

                forwarders {
                        62.81.16.197;
                 150.214.4.34;
                };
        ...
```

Para hacer efectivos los cambios debemos reiniciar el servicio, pero antes conviene ejecutar en un terminal aparte el comando:

```
# tail -f
```

var/log/daemon/ para monitorizar los mensajes de inicio del servidor de nombres. Ahora reiniciamos el servicio:

```
# /etc/init.d/bind9 restart
Stopping domain name service: named.
Starting domain name service: named.
```

10.4 Configuración de un dominio simple

A continuación mostraremos paso a paso cómo configurar el servidor de nombres para definir la traducción directa e inversa de un hipotético subdominio *.linux.aula* que corresponde al rango de dirección IP 10.1.15.0/24. Se trata de una configuración operativa pero intencionadamente no realista, ya que el dominio *.aula* no existe en la realidad y las direcciones IP escogidas no son válidas en Internet. Debemos tener en cuenta que para definir un dominio hemos debido obtener los parámetros del mismo de la autoridad de gestión de la red correspondiente. No obstante, el ejemplo es plenamente válido para una red *privada* que definamos en un entorno que no tenga una conexión directa con Internet.

10.4.1 Tabla directa

Primero añadiremos una sección al final de `/etc/bind/named.conf` para definir nuestra "zona":

```
// add entries for other zones below here

zone "linux.aula" {
```

```
        type master;
        notify no;
        file "/etc/bind/db.linux.aula";
};
```

Aquí indicamos que estamos definiendo una nueva zona, correspondiente al subdominio *linux.aula* de la que somos el servidor de nombres principal (`master`). El parámetro `notify no` indica que no debemos informar a los servidores raíz de esta zona, ya que no es una zona real autorizada en Internet. Si lo fuera, quitaríamos esta línea. En la última línea se indica el nombre del fichero donde se guarda la tabla de correspondencias directa, esto es, de nombres a direcciones IP. El nombre del fichero es arbitrario, pero es conveniente seguir ciertos convenio.

Para construir el fichero `/etc/bind/db.linux.aula` podemos partir de una copia del fichero `/etc/bind/db.local`, que es uno de los que incluye el sistema por defecto para definir una de las zonas estándar, en este caso la que se refiere al nombre *localhost*. Haciendo algunas modificaciones, nos queda:

```
;
; BIND data file for zone linux.aula
;
$TTL    604800
@       IN      SOA     ns.linux.aula. root.linux.aula. (
                             1         ; Serial
                         604800        ; Refresh
                          86400        ; Retry
                        2419200        ; Expire
                         604800 )      ; Negative Cache TTL

                NS      ns                  ; servidor de nombres
                MX      10 mail             ; MTA primario
                MX      20 mail.us.es.      ; MTA secundario

localhost       A       127.0.0.1
gw              A       10.1.15.1
                TXT     "gateway"
ns              A       10.1.15.2
mail            A       10.1.15.3
www             CNAME   ns      ; alias para ns
labtec25        A       10.1.15.25
labtec26        A       10.1.15.26
                MX      10 labtec26
                MX      20 mail
labtec27        A       10.1.15.27
labtec28        A       10.1.15.28
...
```

Como puede verse, lo escrito tras el símbolo `;` es tratado como comentario. La primera línea `TTL` es obligatoria para todos los ficheros de zona. El resto de líneas describen diversos registros en el servidor de nombres. El primero, SOA, es obligatorio y contiene información administrativa como el nombre del servidor de nombres (`ns.linux.aula`), la dirección de correo electrónico del administrador, que aparece como `root.linux.aula` pero que debe interpretarse como *root@linux.aula*, y diversos tiempos estándar para coordinarse con otros servidores de nombres.

Cada definición de registro comienza con un nombre de dominio. `@` se sustituye por el dominio a que corresponde el fichero. Si se omite el nombre en una línea, se toma el de la línea anterior, así como el `IN`, de ahí que no aparezcan en las definiciones de los registros NS y MX. Es importante saber que los nombres de dominio que no acaben en `.` se completan con el subdominio correspondiente al fichero. Así, `mail` se convierte en `mail.linux.aula.`. Ésta es una causa frecuente de error, puesto que un nombre como `mail.us.es` se convertiría a `mail.us.es.linux.aula.` si olvidamos el punto final.

El registro `NS` define el nombre del servidor de nombres para que valga `ns.linux.aula.`.

Los registros MX definen los nombres de las máquinas que se encargan del correo electrónico de este dominio. Se definen dos máquinas, una `mail.linux.aula.` con prioridad 10 y otra `mail.us.es.` con prioridad 20.

Los registros más comunes son los A, que definen la dirección IP correspondiente a cada nombre. No hay que olvidar los nombres que se han usado más arriba para el propio servidor de nombres y para el servidor de correo.

El registro TXT asigna una cadena de texto a un nombre, en este caso a `gw.linux.aula.` para hacer constar alguna información que pueda ser conveniente.

También vemos que es posible asignar más de un nombre a una misma dirección IP mediante un registro CNAME, que define `www.linux.aula.` como un alias a `ns.linux.aula.` y por tanto le hace corresponder la misma IP 10.1.15.2. Hay que tener en cuenta que la definición de un registro CNAME no puede ser otro registro CNAME.

Finalmente vemos como es posible asignar un gestor de correo secundario a una máquina concreta mediante registros MX.

Tras reiniciar el servidor, podemos comprobar si funciona nuestra configuración:

```
# host -a labtec27.linux.aula
labtec27.linux.aula  A 10.1.15.27
# host -a labtec26.linux.aula
labtec26.linux.aula  A 10.1.15.26
labtec26.linux.aula  MX 20 mail.linux.aula
labtec26.linux.aula  MX 10 labtec26.linux.aula
```

10.4.2 Tabla inversa

Para completar nuestra configuración es necesario definir la tabla de conversión inversa, de direcciones IP a nombres. Para ello, añadimos a `/etc/bind/named.conf` la siguiente definición de zona:

```
zone "15.1.10.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/db.10.1.15";
};
```

Como puede verse, las direcciones IP se tratan como si fueran nombres pertenecientes al dominio especial *.in–addr.arpa*. Estas líneas definen una zona para las direcciones de la forma 10.1.15.X, donde las componentes de la dirección IP figuran en orden inverso para ser coherentes con el orden jerárquico que requieren los nombres de dominio. La tabla de correspondencias se guarda en el fichero `/etc/bind/db.10.1.15`, que puede construirse a partir de `/etc/bind/db.linux.aula` ya que comparten el mismo registro SOA. En nuestro ejemplo queda de la siguiente forma:

```
;
; BIND data file for zone linux.aula (inverse)
;
$TTL    604800
@       IN      SOA     ns.linux.aula. hostmaster.linux.aula. (
                             1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

                NS      ns.linux.aula. ; ojo. "linux.aula." no es la zona

1               PTR     gw.linux.aula.          ; ojo con el "." final
2               PTR     ns.linux.aula.
3               PTR     mail.linux.aula.
25              PTR     labtec25.linux.aula.
26              PTR     labtec26.linux.aula.
27              PTR     labtec27.linux.aula.
```

La única novedad con respecto a la tabla directa es que ahora se emplean registros PTR para hacer corresponder las direcciones IP a los nombres. Hay que tener en cuenta que ahora la zona es `.15.1.10.in-addr.arpa.` por lo que hay que escribir los nombres de dominio completos y terminados en ".", mientras que las direcciones IP pueden abreviarse a sólo la componente que falta.

Una vez salvado el fichero y reiniciado el servidor de nombres podemos comprobar la configuración:

```
# host 10.1.15.27
Name: labtec27.linux.aula
Address: 10.1.15.27
```